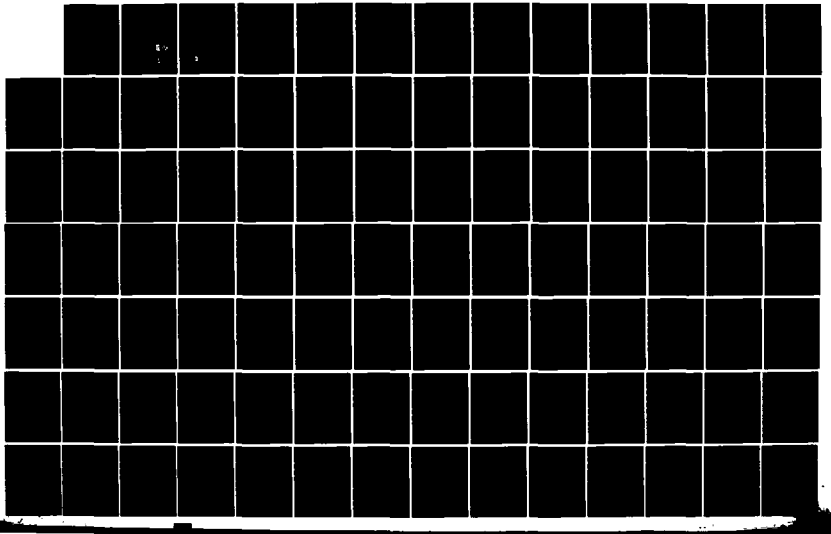
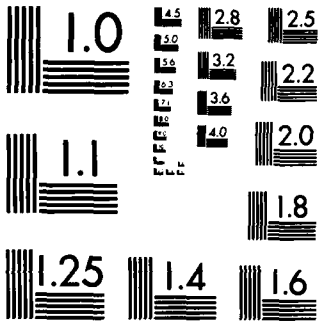
AD-A155 173 THE EFFECT OF COMMUNICATION-LANGUAGE DESIGNS ON SOFTWARE SPECIFICATIONS(U) PERFORMANCE MEASUREMENT ASSOCIATES INC VIENNA VA E M CONNELLY ET AL. APR 85 1/2

UNCLASSIFIED PMA-85-1 N00014-82-C-0499 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

TECHNICAL
REPORT

THE EFFECT OF COMMUNICATION-LANGUAGE DESIGNS ON SOFTWARE SPECIFICATIONS

PMA

TECHNICAL
REPORT

THE EFFECT OF COMMUNICATION-LANGUAGE DESIGNS ON SOFTWARE SPECIFICATIONS

Edward M. Connelly
Kevin J. Garbelman

This research is supported
by Engineering Psychology Group,
Office of Naval Research

April 1985

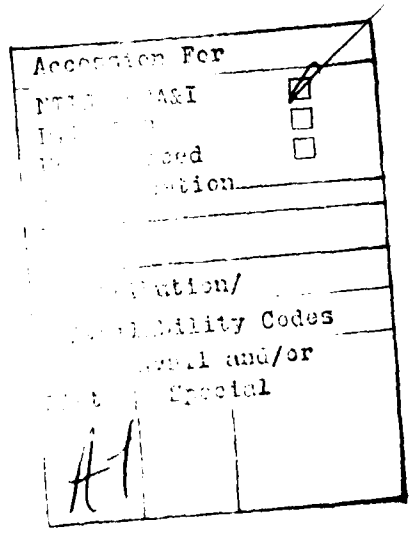
Report Number
85-1

PMA

1909 Hull Road
Vienna, Virginia 22180
(703) 356-1144

PREFACE

The authors are grateful to Michael Olshausen for dedication to his task of editing throughout all revisions, to Coleen Coyle for her work in organizing and analysing the data, and to Susan Baumgardner for her efforts in typing the many revisions of the text. Finally, the authors are grateful to Dr. John J. O'Hare for his support and interest in the effort.



Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Rep. 85-1	2. GOVT ACCESSION NO. AD-A155 173	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE EFFECT OF COMMUNICATION-LANGUAGE DESIGNS ON SOFTWARE SPECIFICATIONS		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Edward M. Connelly Kevin J. Garbelman		8. CONTRACT OR GRANT NUMBER(s) N00014-82-C-0499
9. PERFORMING ORGANIZATION NAME AND ADDRESS Performance Measurement Associates, Inc. 1909 Hull Road Vienna, VA 22180		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153N 42; RR04209; RR0420901; NR 196-175
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research (Code 442) 800 N. Quincy Street Arlington, VA 22217-5000		12. REPORT DATE April 1985 13. NUMBER OF PAGES 99
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Engineering, Software Experiments, Requirements Specifications, Software Human Factors, Cost-Effective Specifications, Non-Programming Users

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The development of complete and accurate software-requirements specification is acknowledged to be a critical factor in the development of software that solves a customer's software problems. The development process usually involves the interaction of one or more customer representatives (users) with one or more software engineers. Among the difficulties frequently experienced in this interaction, one is that the users and the software engineers are each familiar with a different language - the jargon

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

common to their respective fields, but not common to both.

The research described here was an original investigation of the effect of various language designs on the interaction of a non-programming user (experienced in inventory-control systems) with an experienced software engineer, as they endeavored to develop requirements specifications in a cooperative manner. The variations in language design that were studied included: (1) the introduction of collective terms into an existing language, and (2) the use of abbreviations.

Results showed that for the ad hoc, short term interactions studied here, the introduction of collective terms could actually increase the time taken to use all the language terms and could also increase the time needed to detect errors. Further, although the use of abbreviations decreased the time to use terms that had a large number of characters, the time required to employ terms that had just a few characters increased. Apparently, then, the use of abbreviations may not always provide a reduction in the time needed to use the abbreviated terms that is proportional to the reduction in the number of characters in that term.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES

Table 2

Experimental Design:
Repeated-Measures Latin Square

	A_1B_j	A_2B_j	A_3B_j
G_1	A_1B_2	A_2B_3	A_3B_1
G_2	A_1B_1	A_2B_2	A_3B_3
G_3	A_1B_3	A_2B_1	A_3B_2

A_i is a level of problem-complexity.

B_j is a language level.

Each combination of A_i and B_j is an experimental cell.

G_k is a group of 10 participants.

METHOD

Experimental Design

The experiment used a repeated-measures Latin square design, plan 5, cited by Winer (1971). The design is shown in Table 2. The factors investigated were: A) Three levels of problem-complexity, where each level required a different number of commands to complete the task, and B) Three levels of language complexity, where each language level was distinguished by the presence or absence of specific language innovations, which in turn resulted in different amounts of language coding.

Table 3 gives the experiment plan. The numbers in column 1 are the identification numbers assigned to participants, and the hyphenated numbers in columns 2, 3, and 4 refer to the problem-complexity number and the language level, respectively. Session 1 was a pretest. The numbers in column 5 refer to the groups to which the participants were assigned and in column 6,to the order of the problem presentation.

The Participants

The participants were individuals experienced in one or more inventory-control systems, but were not experienced software analysts or programmers. All participants were obtained via newspaper ads. Of the thirty participants, six were female and the rest were male. Twenty-six of the participants had a college degree, and four individuals had a high school diploma. The average age of the participants was 34 years; the youngest being 23 and the oldest 59. Work backgrounds were: managers (25%), computer operators (12.6%), management analysts (9.4%), sales (21.8%), administrators (15.5%), and other (15.5%).

Procedure

Participants were scheduled for either a morning session, beginning at 8:00 a.m. or an afternoon session, beginning at 1:00 p.m. On arrival, each participant was asked to fill out a biographical questionnaire to verify that the participant's experience satisfied the experimental criteria and to obtain additional information regarding the level of experience of each participant in his or her particular field. If a participant satisfied the entrance criteria, the participant was

Table 1

Average Information Transmitted By The Experiment Language Levels

	Language Level		
	1	2	3
Number of Distinct Characters	26	27	19
IMAX (Maximum Possible Average Information)	4.70 BITS	4.75 BITS	4.24 BITS
Average Information with the Zero Order Constraint	3.83 BITS	3.82 BITS	3.47 BITS
Average Information with the 1st Order Constraint	.894 BITS	1.21 BITS	1.76 BITS
Percent Redundancy	80.9%	74.7%	58.8%

Table 1 gives the values of N, IMAX, RC0 (zero order constraint), RC1 (first order constraint), and percent RC1 for each language level used in the experiment. IMAX was calculated by letting N equal the number of distinct characters at each language level. The remaining factors were calculated from the data from all the participants who used each language level.

$$I = \sum_{i=1}^{N} P_i \log_2 \frac{1}{P_i} \qquad (2)$$

In the experiment, the structure itself of each language level implies that there will be an inefficient use of that language level from a theoretical viewpoint (although perhaps not from a practical viewpoint). Consequently, the average amount of information actually transmitted was expected to be less than the theoretical maximum.

The average amount of information actually transmitted by each term in a language can be determined by calculating the probability of occurance of each term, taking into account the constraints of the language design. One constraint, zero order, assumes that each succeeding term in a sequence in the language is dependent only on the distribution of terms and not on the terms previously received. Another constraint, first order, assumes that the probability of a term is dependent on the previous term received (or, equivalently, that the probability is dependent on the distribution of each combination of two terms.) Yet another constraint, second order, assumes that the probability of a term is dependent on two previous terms (or, equivalently, that the probability is dependent on the distribution of each combination of three terms), and so forth, for higher levels of dependency. Of interest to this experiment were the zero and first order constraints only, for their ability to indicate the average amount of information transmitted by each language level.

Related to the amount of information transmitted by a language is the amount of information redundancy and the percent of information redundancy. If we define IMAX as the theoretical maximum average amount of information transmitted (for the case in which the term probabilities are equal) and ICX as the actual average amount of information transmitted, where CX represents an X-order language constraint, then the information redundancy for that level of constraint is given by:

$$RCX = IMAX - ICX. \qquad (3)$$

The percent of information redundancy is given by:

$$\%RCX = 100\left[\frac{IMAX-ICX}{IMAX}\right] \qquad (4)$$

Abbreviated terms and unabbreviated terms can both be considered to form codes, with each term having a definite, pre-established meaning. According to information theory, the amount of information transmitted by a code is dependent on the likelihood of the next term giving information contained in the previously transmitted terms. When there is an equal likelihood of the next term being any one of the possible terms defined in a language, the amount of information transmitted is maximized. As structure or constraints are introduced into the language, terms are no longer equally likely, and the amount of information transmitted is reduced.

The objective of this experiment was to determine if there was a significant relationship between the theoretical efficiency of the language levels employed in the experiment, as measured by the amount of information they transmitted, and their actual efficiency as measured the time required by participants to solve the experiment problems using those language levels. In order to formulate an Information Measure, we regarded the nonprogramming user as a transmitter of information (i.e., transmitting requirements' needs), while the software expert was regarded as a receiver of this information. If a language is designed to transmit the maximum amount of information, it has no redundancy, and, consequently, every possible sequence of terms is a valid message. In that case, there is no way of detecting an error by examining a sequence of terms, as received. At the other extreme, a language design that results in the transmission of little information has considerable redundancy. In that case, many terms must be transmitted to communicate a message. We note that the English language, depending on its specific use, is estimated to transmit 30% to 50% of the maximum possible information. If we, as human beings, have become accustomed to that level of redundancy or if that level of redundancy has evolved because it is best for humans to use, the relationship between the theoretical efficiency of a synthetic communications process and its actual efficiency, for example in generating requirements specifications, cannot be expected to be a linear relationship: there may be too much, or too little, redundancy in the synthesized language.

If there are N terms available in a language, and each is equally likely, then knowledge of the terms previously received provides no information about which terms will be received next. In that case, the likelihood of receiving each term next is:

$$P = 1/N. \qquad (1)$$

For a language in which all the terms are coded for maximum efficiency, the average amount of information transmited, known as I, is given by Shannon's information-theoretic measure:

terms refer to all the parts of a car -- the nuts, bolts, windows, panels, etc., then higher level terms could refer to assemblies such as "a motor," "a frame," and so forth. It is easy to envision even higher level terms that are combinations of lower level terms. Thus, the "car" itself can be defined as including a motor, a frame, etc. For any application, it is possible to develop a set of terms by means of which any item or collection of items can be efficiently identified.

One problem, however, is that the effect on the quality of the communication process of introducing new terms often is not clear. Should every item be referred to by a unique term? Or, would some items be more effectively referred to by a heirarchy of terms? For instance, and using the car as an example again, a wheel can be referred to as a "wheel" or, alternatively, as a "tire, rim, and valve." The second alternative uses three lower-level terms and so avoids, but somewhat absurdly, introducing the higher-level term. In this case, because the term "wheel" is already in our vocabulary, we can use it without introducing it as a new term; however, there are many opportunities in a conversation between two experts from different fields to create new, ad hoc, higher-level terms. And, if such terms are too frequently introduced, the conversation may become unwieldy, i.e., the participants may have difficulty trying to recall the terms' meanings. Thus, tradeoffs may exist between the number and the level of new terms that should be introduced into a conversation in order to aid in the communication process. The research reported here was an investigation of the effect of various language innovations, specifically the introduction of collective terms and then, following that, the introduction of abbreviations, on the effectiveness of the special type of communication process (viz., a short meeting between two individuals who are experts in different fields and who are working cooperatively to solve a problem) considered here.

Information Measurement

By introducing the language innovations - the collective terms and the abbreviations - one at a time, three languages or language levels were constructed. Each language level was restricted in the sense that an allowable set of terms was prespecified and only those terms were permitted for communication. Language Level 1 contained only item-specific terms, plus those grammatical terms needed to form commands. Level 2 introduced collective terms. Level 3 retained the collective terms of Level 2, introduced two new commands, and required that all communications be in abbreviated form.

In order to evaluate differences in strategies used by participants employing different communications modes, PMA, Inc. conducted a preliminary evaluation using two modes: terminal, and face-to-face communications between two individuals attempting cooperatively to solve a problem. Communications in those sessions where both participants interfaced via terminal keyboards were recorded directly in the computer memory. Face-to-face session communications were recorded on video tape and later were typed into a computer for analysis. It was found, as Weeks and Chapanis had found, that the number of communications and the number of words used were both far greater in the face-to-face sessions. It was also found that the nature of the communication in the two modes was quite different. Sentences were structured in a more formal way when using the keyboard. Also, not only were more communications as well as more words used in face-to-face communications, but the actual words used were different.

But of special interest to this investigation was that the strategies used by the participants were quite different in the two modes. For instance, with the terminal, the text of the immediate past (up to 24 lines of text) from both parties was available for review on individual screens, so that participants did not have to remember that previous information. At post-test interviews, the participants stated that the display text was used often and was important to their problem solving. Our conclusion was that the two modes resulted in distinctly different communications in several important ways, including both different syntax and vocabulary and different problem-solving strategies. Consequently, it was concluded that one experiment design could not be produced to study behavior in both modes. As a result, an experiment was designed to use terminal communication. The results obtained should be attributed to that mode of communication only and should not be attributed to problem solving involving face-to-face communications.

Discussion of Variables

Developing a Working Vocabulary

When a non-programming user and a software expert meet to develop RS, a working vocabulary consisting of mutually understood terms must be established. At the lowest level of vocabulary complexity there will be terms representing each possible item that can be referred to in the discussion. At a higher level, terms may be defined that refer to collections of lower level items. For instance, if the lowest level

themselves, and perhaps multiple interchanges among the several designers. At present, when specifications for the development of a large software system are considered, the user-client develops formal specifications without extensive interchange concerning the ultimate designs. The RS are then presented to designers, perhaps in the form of a request-for-quotation. Such a procedure, though often used for large software projects, is formal and prohibits the informal interchange described above that might well be used to advantage in the development of a smaller software system.

Literature on Human Communication

The literature on human communication, in general, is voluminous; hence, our interest here was restricted to the interaction of two specific types of communication. The first type was that of cooperative communication between two individuals working together to solve a problem. One restriction was that two individuals had to be experts in different areas but could not be experts in each other's areas. (In the experiment, one participant was an expert in inventory control, but was not a software designer or coder; the other participant was an expert software engineer, but had only superficial knowledge of inventory control). The second type of communication was that of specialized or expert "jargon," subject to the restriction that the jargon was of an ad hoc nature, i.e., that the communications involved only one-time, short-term interactions. The one-time, short-term restriction tended to limit the number of specialized terms or abbreviations that could be used effectively without forcing participants to frequently look up definitions in a dictionary.

Weeks and Chapanis (1975), described experiments in which participants worked cooperatively on some problems and competitively on other problems using four communication modes: face-to-face, closed circuit TV, telephone, and teletypewriter. Their conclusions regarding cooperative tasks, which was the area of interest here, were: (1) that problems are solved twice as quickly using a voice mode compared to a typewriter, (2) that two-thirds of the problem-solving time in all modes deals with active communication, (3) that voice modes result in more verbose communications (use of a voice mode results in four times the number of messages and words), and (4) that the communication rates, measured in words per minute, are about eight times greater for voice modes than for teletypewriter modes. Weeks and Chapanis did not report on any differences in the problem-solution strategies that may have been used with the different communication modes.

designer by describing the interchange between the two, and he noted that designers often use the technique of suggesting particular pieces of equipment or procedures that might be (or might at least approximate) an acceptable solution. The client, in rejecting some of these suggestions, modifies the requirements statements. As a result of this designer–client interchange, the client clarifies his or her understanding of the problem, and, by working together, the client and designer arrive at an acceptable solution.

Miller pointed out that the role of the designer is to provide facts about the real world, in terms of the properties of equipment and alternative solutions, as well as to ask questions which, while providing clarification, frequently may have the effect of inducing the client to identify a new problem or to better conceptualize the present problem. Miller further identified a sequence of six states which the client and the designer use sequentially. These six states are:

1. Goal statement
2. Goal elaboration
3. (Sub) Solution outline
4. (Sub) Solution elaboration
5. (Sub) Solution explication
6. Agreement on (Sub) solution.

Miller indicated that this state-sequence was used iteratively, but that sometimes the sequence was truncated in order to start a new sequence in the pursuit of a different solution.

The results by Miller suggest that the process of transforming a user's needs into a formal statement of requirements may benefit from the interchange between a client knowledgeable about those needs and a software designer knowledgeable about the capabilities of computer systems. According to this model, the client's concept of need grows as a result of the interchange, until the client becomes aware of new and different possible solutions to the problem. New solutions evolve interactively until a final solution emerges that is accepted by both the client and the designer as being complete and feasible.

The question that arises, however, is: How comprehensive should these interchanges be to evolve a feasible solution? For instance, when preparing RS for a large computer system, it may not be possible for all the user-clients to have a useful interchange with one, or more, of the designers. Not only would there have to be multiple client-designer interchanges, but there would also have to be multiple interchanges (discussions of tradeoffs among the many interests) among the user-clients

a. the goal objectives,
b. the systems or environments involved,
c. the constraints (on performance, delivery, costs, etc.), and
d. the resources available for system-design development.

2. Functional requirements specifications determining precisely what the final product must be like, including:

   a. every important aspect of the product's internal performance,
   b. the characteristics of its embedded operator/user population,
   c. its relationship to other systems and environments, and,
   d. the development constraints.

3. Overall, high-level design translating the functional requirements into a comprehensive design which specifies the major components of the to-be-developed product, and describing for each component:

   a. the goals to be achieved by the component,
   b. the characteristics of all factors to which the component is to be sensitive, i.e., the input,
   c. the characteristics of the effects the component must achieve, i.e., the output,
   d. the internal structures of the component, and
   e. the general principle(s) of any operation sequences within the component information-processing procedures.

4. Detailed design suitable for prototype development.

The steps of interest to this investigation were the first and second, which start with the initial discussions of the problem between the client and software designer(s) and end with preparation of formal RS.

Miller investigated the transformation of a client's vague, initial specifications into precise and formal specifications. He described, in particular, the functions of the client and the software

relationships, and automatically provides analytical summaries, such as problem-statements, directories, hierarchical-structure reports, and graphical summaries of data flow and data relationships. Another example in the class is a method called Software Requirements Engineering Programs (SREP), described by Boehm (1976) in a survey of methodologies. SREP uses the data-management system of ISDOS and, in addition, produces functional simulations from requirements statements. SREP is used for configuration control, traceability from requirements to design, and report generation. Still a further method in the second class, Structured Analysis for Requirements Definition, is described by Ross and Schoman (1977). Part of it is a Structured Analysis and Design Technique (SADT) for analyzing requirements using graphical techniques.

All these methods, despite their complexity, contain a common limitation: that of providing a structure for recording RS and then for analyzing those RS, but not for supporting the process of developing RS from a user's needs.

In an extensive survey and review of the status of software-requirements methods, Ramamoorthy & So (1978) identified the same requirements-development problems referred to above; namely, that a large percentage of the total errors in software development occur in the requirements specifications, and that these errors cause serious problems leading to high costs, unresponsive products, slippage of production schedules, and difficulty in system operation and maintenance. They go on to briefly describe a number of methodologies for RS documentation which they feel may aid in the software-design process.

Research on Development of Requirements Specifications

Miller (1978) investigated the interactive process between a user (client) and software designer in analyzing the user's needs, establishing RS, and developing a software design. His description of the interactive process identified four steps, and is presented below. For ease of reference in what follows, we give all four of Miller's steps, even though our interest here is limited to the first two steps.

The four steps Miller identified are:

1. Problem understanding, arriving at a general agreement as to what are:

1. To identify the capabilities of, and the strategies employed by, software users in specifying a minimum-cost inventory-control system, and to establish a base-line study of presently available problem-solving aids.

2. To evaluate aids that a user and a software expert could use together to build a working vocabulary of software and software-application terms.

3. To evaluate sophisticated aids with which to assist a user in providing more complete RS.

4. To evaluate aids to assist a neophyte user in developing RS.

The results of the first experiment were presented in Connelly (1984) and showed that, "without the benefit of sophisticated aids," the non-programming participants of that study (inventory-control experts) were not able to generate RS "capable of guaranteeing a least-cost system." One conclusion of this 1984 study was that "improvement might have to be directed by a software expert," i.e., that at the core of RS improvement would be the user/expert interaction. The research reported in this present report was an evaluation of two basic techniques, the use of abbreviations (and, correlatively, of acronyms) and the use of collective terms, both commonly employed as aids in building working vocabularies to be used in developing RS. The results, as will become apparent, were both illuminating and somewhat surprising.

Literature Review

Tools for Software Development

At present, the tools available for software development fall into one of two classes. The first includes the well-known design-aids such as Structured Design (Yourdon & Constantine, 1979), Jackson's Method (Jackson, 1975), and Logical Construction of Program (Warnier, 1981, and Orr, 1981). All of these design aids use RS which are assumed to be correct as their starting points.

The second class of aids provides structures with which to record and analyze RS. In this latter class is a system called ISDOS (Teichroew & Hershey, 1979), which uses a problem-statement language (PSL) and a problem-statement analyzer (PSA). ISDOS permits a formal description of a system in terms of entities, classes, and

INTRODUCTION

The initial step in the development of a software product is the development of software-requirements specifications. Frequently, a requirement specification is developed in a specification conference between a non-programming customer (the user) and an experienced software engineer. The effectiveness of that interaction is critical to the development of a requirement specification that reflects all the needs of the customer, and yet is practical (cost effective) to implement. A misunderstanding of what is required, or of what is expensive to implement, may likely not be detected until the software product has been produced and first used by the customer. Unfortunately, to correct errors that are detected only during a program-test can be costly since, at that late stage, error correction may require substantial redesign and recoding of the program.

During the specification conference both the user and the software engineer are faced with many unknowns regarding the feasibility or practicability of possible features of the system. The user may not know, for instance, what features come at almost no cost, and what features require extensive programming effort and computer time to implement. Further, the user is faced with the problem of attempting to develop requirements specifications while working with a software engineer who undoubtedly "speaks a different language." Generally, this results in an interactive process during which the software engineer learns about the application at hand and the user learns something about what is practical to implement. During this process, certain trade-offs are fashioned, sometimes involving many 'cut and try' solutions, which finally result in mutually agreed-upon software-requirements specifications (RS).

Since the quality of the interactive process in the specification conference is critical to the quality of the resulting RS, and since, as discussed in the literature review below, the poor quality of RS is often cited as the principal reason for cost overruns, time delays, and, sometimes, throw-away code, an investigation of possible means for improving the medium of exchange -- that is, the language -- used during that interactive process was undertaken.

This was the second of four previously planned experiments which sought, respectively:

Table 3

Experimental Plan

Participant #	Session 2	Session 3	Session 4	Group	Order
	Problem-Complexity - Language Level				
1	2-3	1-2	3-1	1	5
2	2-2	1-1	3-3	2	5
3	2-1	1-3	3-2	3	5
4	1-2	3-1	2-3	1	4
5	1-1	3-3	2-2	2	4
6	3-1	2-3	1-2	1	3
7	1-2	2-3	3-1	1	1
8	1-1	2-2	3-3	2	1
9	1-3	2-1	3-2	3	1
10	3-3	2-2	1-1	2	3
11	3-3	1-1	2-2	2	6
12	3-2	1-3	2-1	3	6
13	3-1	2-3	1-2	1	3
14	3-3	2-2	1-1	2	3
15	3-2	2-1	1-3	3	3
16	2-3	3-1	1-2	1	2
17	2-2	3-3	1-1	2	2
18	2-1	3-2	1-3	3	2
19	1-2	2-3	3-1	1	1
20	3-2	2-1	1-3	3	3
21	1-3	2-1	3-2	3	1
22	2-3	3-1	1-2	1	2
23	2-2	3-3	1-1	2	2
24	2-1	3-2	1-3	3	2
25	1-2	3-1	2-3	1	4
26	1-1	3-3	2-2	2	4
27	1-3	3-2	2-1	3	4
28	3-1	1-2	2-3	1	6
29	3-3	1-1	2-2	2	6
30	3-2	1-3	2-1	3	6

provided with a consent form and asked to read and sign it. If a participant's experience did not satisfy the entrance criteria, that participant was excused from the experiment.

Each qualified participant was next seated in the experiment room, which contained a video tape recorder, a video monitor, and a computer terminal on a table. All instructions for the experiment were presented via a video tape.

As each participant began the experiment, the individual read a description of the experimental problem that included instructions for solving it as well as an example solution using the language of the pretest. Also included were instructions for the pretest problem. During the experiment, as each of the experimental problems was presented, the instructions for that problem were given to the participant via the video-tape system. In working the problems, the participants used the terminal keyboard to communicate with a software expert who sat in an adjacent room and used his computer terminal to provide the information requested by the participant. The same software expert was used throughout the experiment.

The participants were told that up to one hour (wall clock time) was allotted for each problem and that the computer would automatically stop all work on a problem when that time limit was reached. Participants were permitted to take a short break between test problems, if they so desired.

Experimental Task

The participant's task was to play the role of an inventory-control manager working with an experienced software expert to develop the software-requirements specifications for an inventory-control system. In accomplishing this task, the participant was allowed to use only one of the three language levels constructed for the experiment for each one of the three problems that were presented to him or to her (see Table 3). In reality, the three problems were actually the same inventory-control problem, but presented at three different levels of complexity.

To fully understand the experiment, it is now necessary to discuss, in some detail, the basic inventory-control problem. The experimental inventory-control system had the job of keeping track of parts for a hypothetical factory and consisted of two files and a set of "change-records." The files consisted of an old master file and a new

master file. Each week, the old master file was supposed to be updated according to the change-records to produce a new master file. Since the change records might contain errors, however, it was deemed important to test them prior to their use so that any errors could be detected and corrected, and thus not entered into the new master file. However, the change-record tests were themselves an expense to the factory, since the tests had to be proposed, developed, programmed, used, and maintained. Should a redundant test be specified, funds would be expended without the benefit of the detection of additional errors. Further, if a test designed to detect very rare or nonexistent errors was specified, funds might be expended without a cost-effective benefit. Conversely, if tests needed to detect frequent and potentially-costly errors were not specified, unnecessary costs to the company could accrue, because the undetected change-record errors could result in orders for excess stock or in a failure to order necessary stock, with the attendant production shortages. Thus, change-record tests had to be carefully specified. Necessary, cost-effective tests had to be included; but the ineffective tests had to be eliminated.

For experimental purposes, the cost-effectiveness of each test was represented by points assigned according to how effectively that test detected errors for each change-record. For instance, as shown in Figure 1, which is similar to the screen presented to the participant, Test 4 was assigned 10 effectiveness points for change-record 1, and 8 effectiveness points for change-record 9. In the Figure, only data for change-records 1 and 9 (CR1 and CR9) are shown; the actual screen presented data for all the CR's (9 total). If more than one test was applied to a CR, the total of the effectiveness points of all the tests was presented on the screen. In Figure 2, where Test 1 was added to Test 4, a total of 15 points accumulated for change-record 1 and a total of 12 points for change-record 2.

Each change-record required a prescribed minimum number of effectiveness points before the set of tests selected for it by the participant became effective in detecting errors. If, for a given change-record, the sum of the points for all the tests assigned by the participant was less than that minimum number, additional tests had to be requested, until the point total equaled or exceeded the minimum. It was possible, however, to have too many total effectiveness points per change-record. This occured when two or more tests detected the same error, a redundancy that would result in an unnecessarily costly system. To limit this possibility, a maximum number of points was also specified for each change-record. If the total number of effectiveness points exceeded that maximum number, one or more of the tests had to be deleted, thus reducing the total number of points. For instance, in Figure 3,

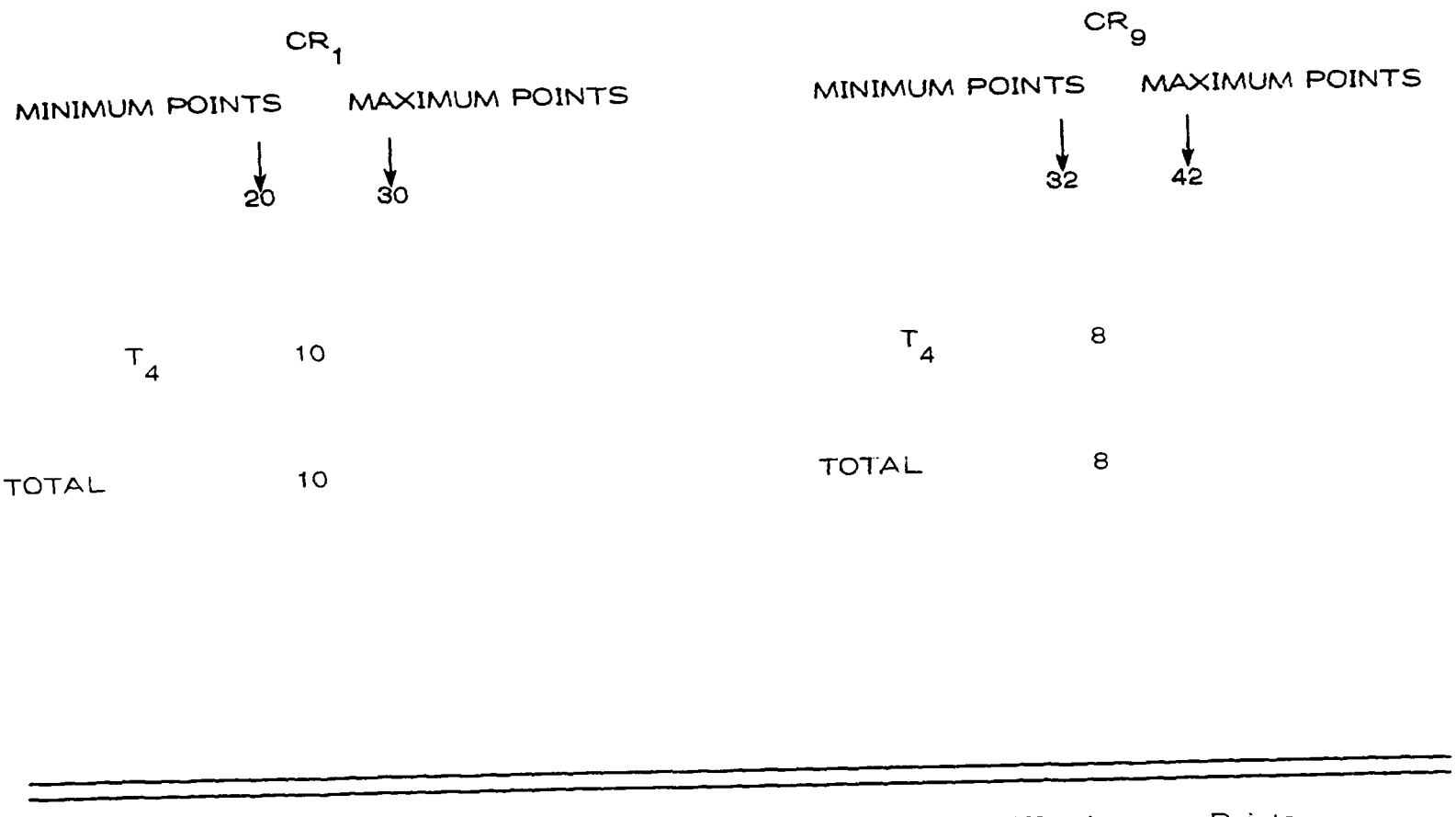


Figure 1. Variation In Change-Record Effectiveness-Points Between CR's For a Single Test

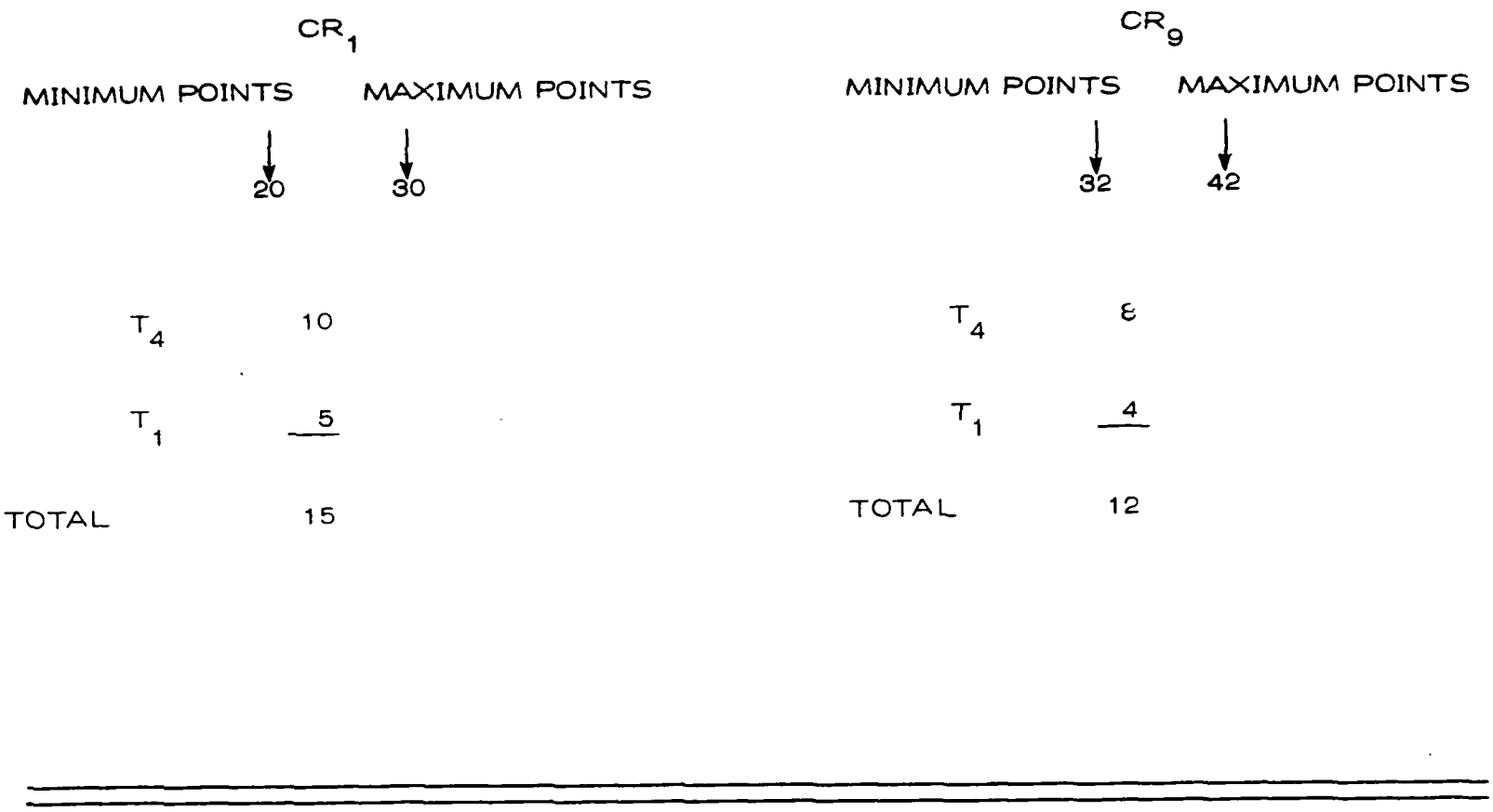


Figure 2. Summation of Change-Record Effectiveness-Points For Two Tests

CR_1		CR_9	
MINIMUM POINTS ↓ 20	MAXIMUM POINTS ↓ 30	MINIMUM POINTS ↓ 32	MAXIMUM POINTS ↓ 42
T_4	10	T_4	8
T_1	5	T_1	4
T_6	11	T_6	12
T_{10}	10	T_{10}	10
TOTAL	36	TOTAL	34

Figure 3. Summation of Change-Record Effectiveness-Points: Excessive vs. Acceptable Sums

Tests 4, 1, 6, 10 have been assigned to change-record 1, for a total of 36 effectiveness points, although the maximum number of points is actually 30. Consequently, one or more of the tests must be deleted. Note, however, that these same tests when applied to CR9 result in a total of 34 points, which is within the acceptable range for CR 9. For each of the 9 change-records, the tests were to be assigned by participants so that the total number of effectiveness points was greater than the minimum, but less than or equal to the maximum. There was a total of 225 tests from which the software expert could choose in response to commands entered by the participants.

Different ranges for the minimum and maximum number of effectiveness-points were chosen in order to establish the three levels of problem complexity called for in the experiment design. Figures 4, 5, and 6 give the ranges for the minimum number of acceptable effectiveness points for problem complexities 1, 2, and 3, respectively. For complexity-level 1, the minimum ranged from 9 to 12 points (the actual value within the range was selected by the computer both automatically and at random at the beginning of each session so that, as discussed in a subsequent paragraph, the expert software engineer would not know initially the specific minimum and maximum effectiveness points.)

For problem-complexity-levels 2 and 3, the ranges were linked to different groups or "Sections" of CR's, as shown in the respective Figures. These coordinated ranges made it advantageous for the participant to use various section commands - sometimes requesting tests to increase points for many CR's, such as with the "Section 7" command, and yet at other times requesting tests that were effective for only a few CR's (see Figure 7).

Figure 8 gives the maximum effectiveness points for all the problem-complexities. An increment of 3, 4, or 5 was selected at random, each increment with an equal probability, and that increment was then added to the minimum point value to establish the maximum value. As with the minimum value, the maximum value was determined automatically within the computer at the beginning of each session so that it was unknown initially to the software expert.

To repeat, for the sake of emphasis, the same software expert was utilized in each experiment, while the participant (an expert in inventory-control systems) was a different individual in each experiment. The participant issued requests, actually commands, to the software expert exclusively by means of the pre-established vocabaulary of the language level in use. The software expert, who was familiar with the various CR tests, received these commands from the participant and

Minimum Effectiveness Points For
Problem Complexity-Level 1

For CR_1 through CR_4

Minimum Value	Probability of Minimum Value
9	.25
10	.25
11	.25
12	.25

Mean = 10.5

Figure 4. Minimum Effectiveness-Points For Problem Complexity-Level 1

Minimum Effectiveness Points For
Problem Complexity-Level 2

For CR_1 through CR_3

Minimum Value	Probability of Minimum Value
26	.25
27	.25
28	.25
29	.25

Mean = 27.5

For CR_4 through CR_6

Minimum Value	Probability of Minimum Value
30	.25
31	.25
32	.25
33	.25

Mean = 31.5

Figure 5. Minimum Effectiveness Points For Problem Complexity-Level 2

For CR_7 through CR_9

Minimum Value	Probability of Minimum Value
34	.25
35	.25
36	.25
37	.25

Mean = 35.5

Figure 5. (Cont.) Minimum Effectiveness-Points For Problem Complexity- Level 2

Minimum Effectiveness-Points For
Problem Complexity Level 3

For CR_1 through CR_3

Minimum Value	Probability of Minimum Value
44	.25
45	.25
46	.25
47	.25

Mean = 45.5

For CR_4 through CR_6

48	.25
49	.25
50	.25
51	.25

Mean = 49.5

For CR_7 through CR_9

52	.25
53	.25
54	.25
55	.25

Mean - 53.5

Figure 6. Minimum Effectivenss Points For Problem Complexity Level 3

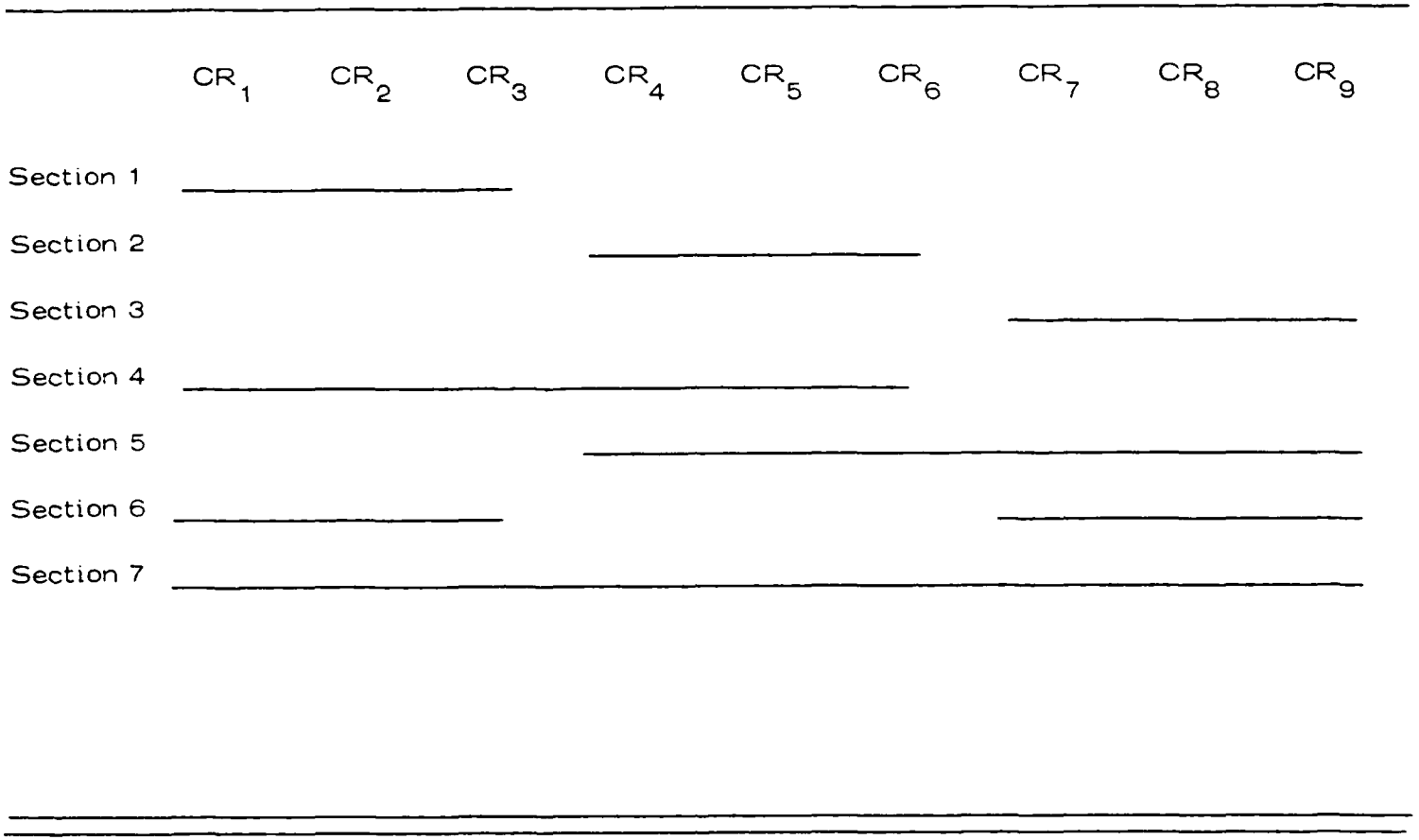


Figure 7. CR's Referenced By Each Section Command

Maximum Effectiveness-Points For
All Problem Complexities

Maximum = Minimum + Effectiveness-Point Increment

Effectiveness Point Increment	Probability
3	.33
4	.33
5	.33

Mean = 4.0

Figure 8. Maximum Effectiveness-Points For All Problem Complexities

assigned whichever test most closely fit the commands entered. The software expert knew, in general, what the task requirements were, but he did not know the specific end objective -- in particular, he did not know the minimum and maximum point limits for each CR. If questions arose regarding the conduct of the experiment, the participant was allowed to seek clarification by communicating with the software expert in standard English, but only through their mutual CRT keyboard interfaces (Figure 9). The participant's task was to solve each problem by issuing commands using the pre-established vocabulary of the language level that had been assigned to the problem on which he or she was working.

Language Levels

Pretest Level

This language level, used only for the pre-test, permitted the following commands:

ADD _ _ POINTS TO CR _ .

ADD MORE THAN _ _ POINTS TO CR _ .

ADD LESS THAN _ _ POINTS TO CR _ .

DELETE TEST _ _ _ .

AND (to form compound commands)

*** (error, delete this line).

Note that dashes represent the numbers (one dash per digit) input by the participant to specify points, CR's, or tests. Note also that, at this language level, as at all other levels, commands could be compounded using the conjunction "AND." For example:

ADD MORE THAN _ _ POINTS TO CR _ AND DELETE TEST _ _ _.

Language Level 1

Language Level 1 permitted the following commands:

ADD _ _ POINTS TO CR _.

ADD _ _ POINTS TO CR _.

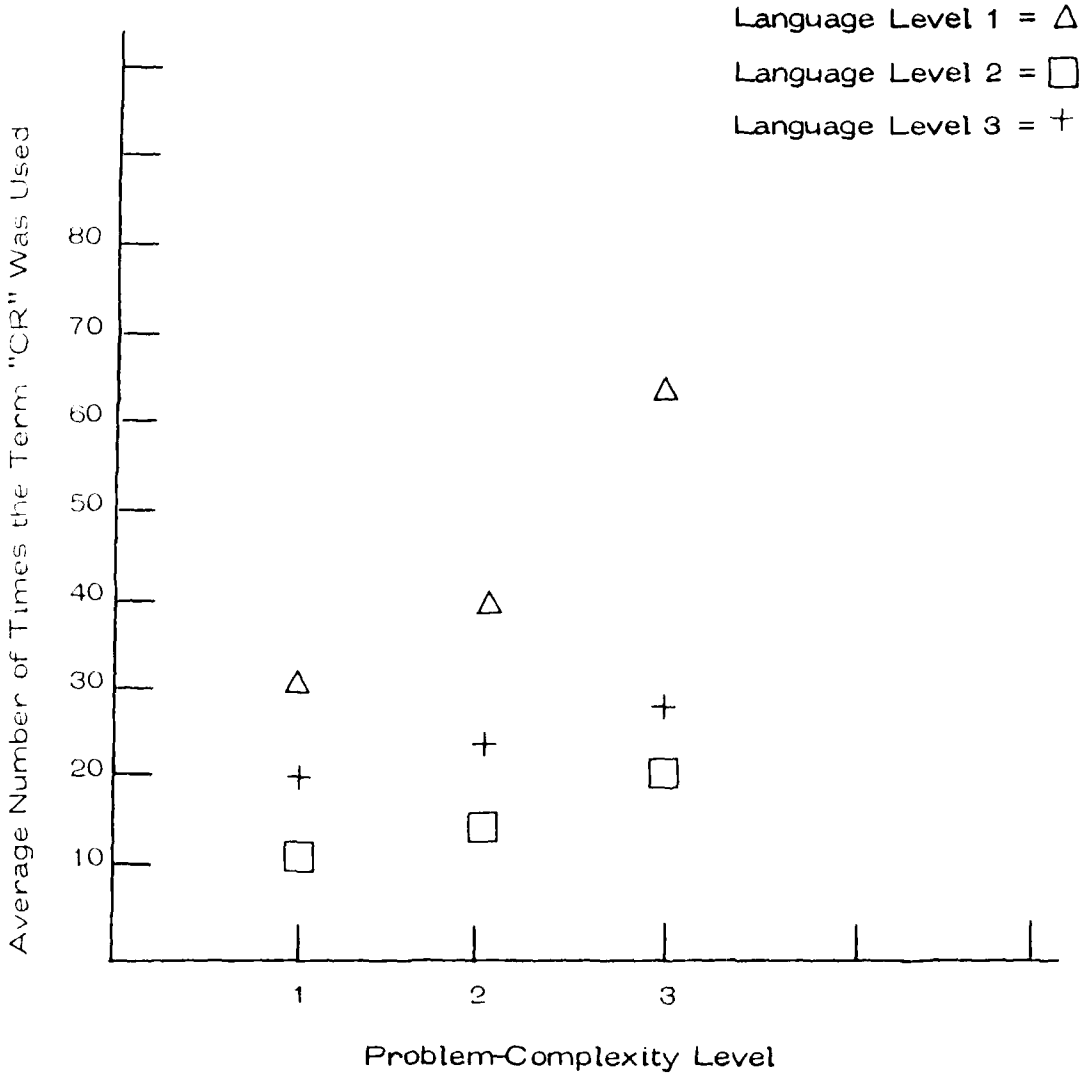


Figure 13. Average Number Times the Term "CR" Was Used To Solve the Total Problem

Figures 13 and 14 show, respectively, the number of times the terms "CR" and "Points" were used. For both terms, usage increased with increasing problem complexity. However, as a function of language level, the greatest use of CR's occured at Language Level 1, followed by a lesser use (of implied) CR's at Language Level 3, followed by the least use of CR's at Language Level 2. The reduction between Language Levels 1 and 2 can be explained by the introduction of the collective term "Section" at Language Level 2, and was expected. But, the increase in the use of (implied) CR's at Language Level 3 over the use of CR's at Language Level 2 was not expected. This suggests that the participants employed a different strategy when working with Language Level 3 than they did when working with Language Level 2.

Similar use-strategy was found for the term "Points". Thus, the decrease in the number of times the term "Points" was used at Language Level 2 compared to Language Level 1 could be explained by the introduction of the collective term "Section". Fewer total commands (because "Section" commands were more efficient) were expected; hence, the term "Points" would be used less frequently. But, again, use of Language Level 3 resulted in an increase in the use of the (implied) term "Points" over Language Level 2. Since the term "Points" occured in CR commands as well as in "Section" commands, this increased use at Language Level 3 apparently resulted from an increase in the use of CR commands over Section commands, compared to Language Level 2, once again suggesting that these two language levels required different strategies.

Results for the number of keystrokes per line are given in Figure 15. As expected, the number of keystrokes decreased significantly with increasing language level. A moderate decrease in the number of keystrokes occured between Language Levels 1 and 2, with a sharper decrease occuring in the move to Language Level 3. This reflected the sequential increase in keystroke efficiency that resulted from the introduction, first, of the more powerful "Section" commands and then, second, of the abbreviations for the entire commands.

Results for the measures "average number of keystrokes to solve the total problem, "... to solve the transiting portion", and "... to solve the homing portion" are given in Figures 16, 17, and 18, respectively. In each figure, the number of keystrokes decreased consistently as the language level increased, as was expected. The introduction of the collective "section" command, begun at Language Level 2, permitted the participant to solve the experimental problem with fewer keystrokes than

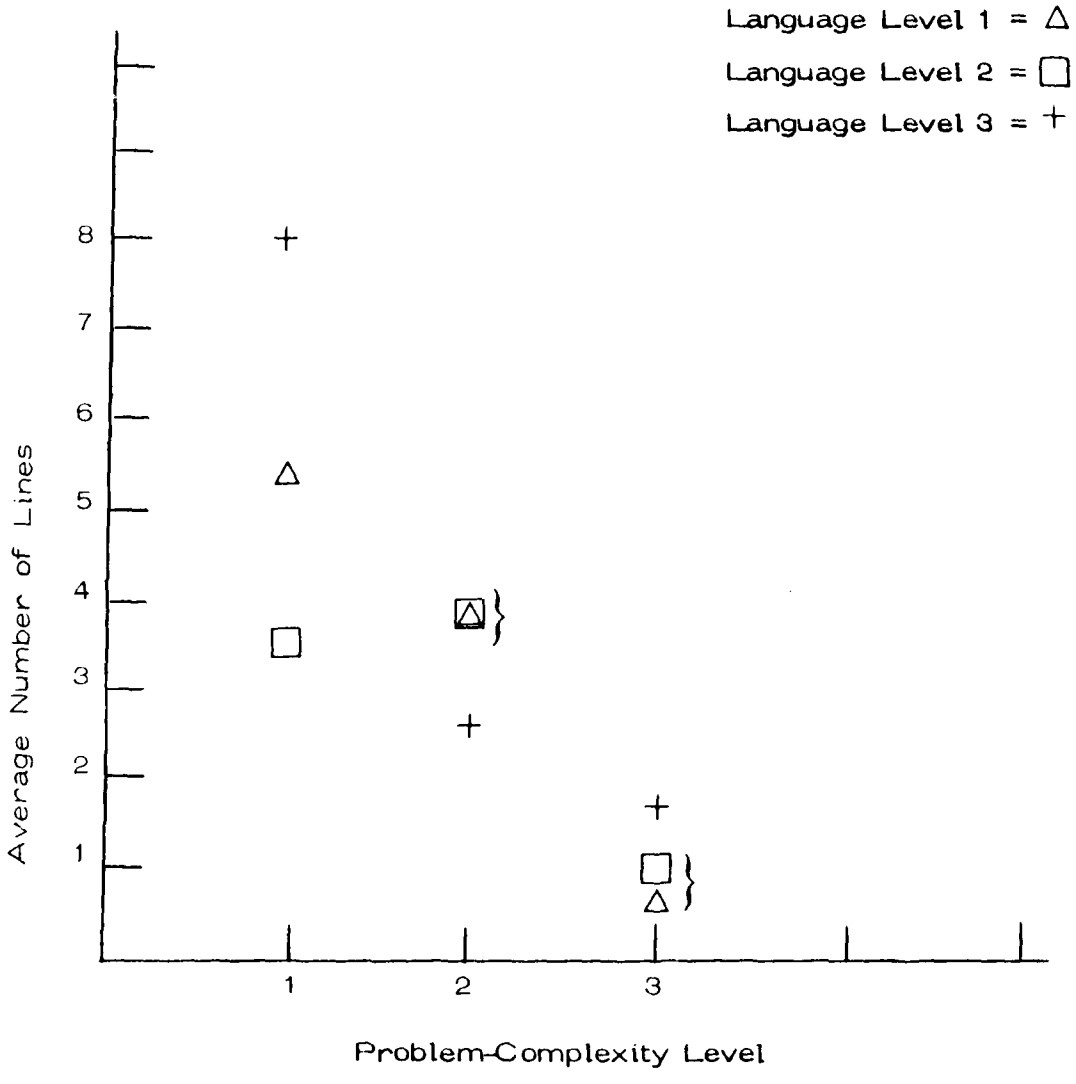


Figure 12. Average Number of Command Lines To Solve the Homing Portion

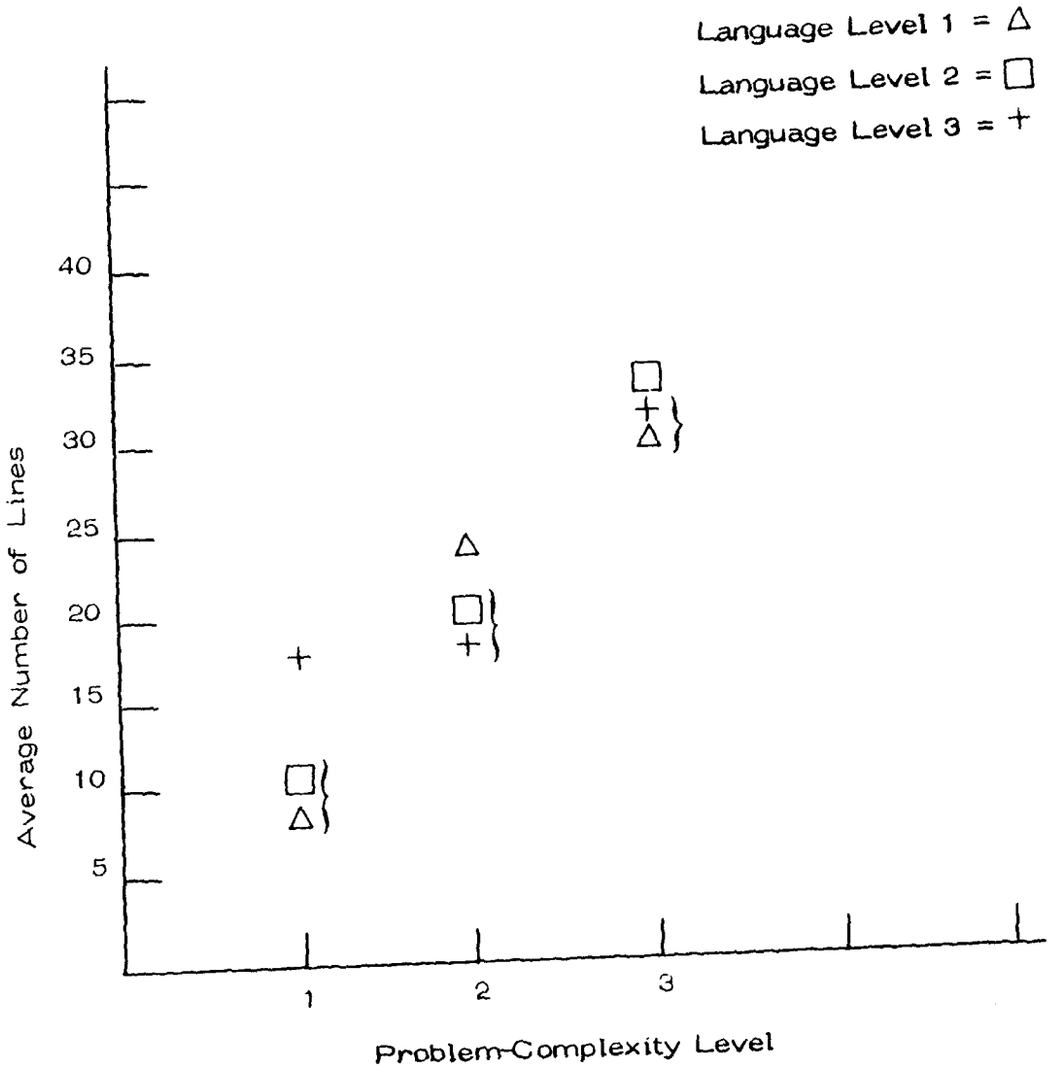


Figure 11. Average Number of Command Lines To Solve the Transiting Portion

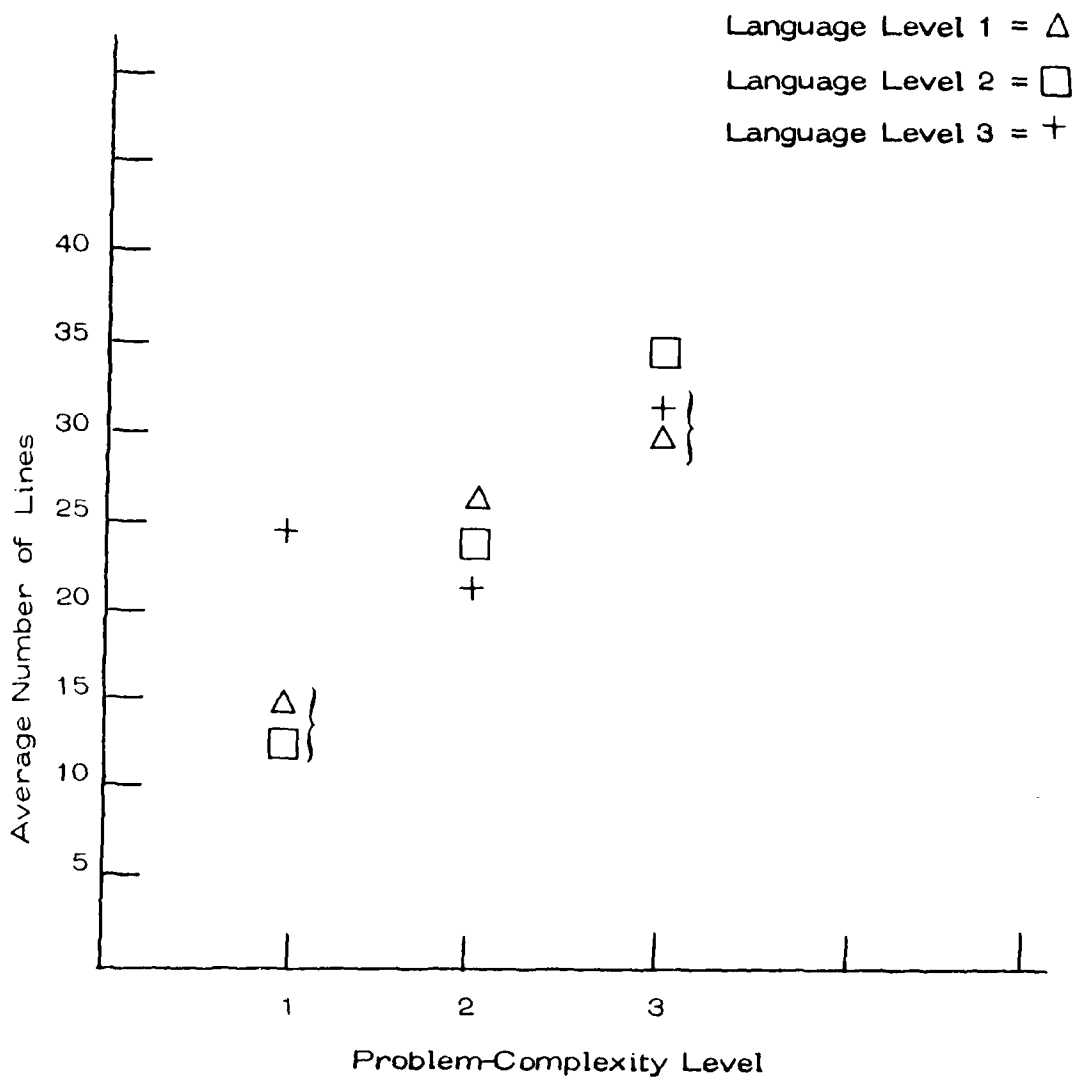


Figure 10. Average Number of Command Lines To Solve the Total Problem

Analysis 3: Student-Newman Kuels Test

Purpose

To evaluate the significance of the experimental factors and of the experiment cells on the performance measures.

Method

The Student-Newman Kuels (SNK) (Sokal and Rohlf, 1969) posterior comparison-of-means test was used to evaluate the significance of the effects of the factor levels and the experiment cells on the thirteen performance measures.

The results are shown in Figures 10 through 22, in which the language-level effects have a statistical significance of at least .01. In each figure, a bracket enclosing plot symbols at each problem-complexity level identifies language-level effects that were not significantly different. Thus, in Figure 10, at problem-complexity level 1, the number of command lines required to solve the total problem was not significantly different for Language Levels 1 and 2, although the number of lines required at Language Level 3 was significantly higher than at either Language Level 1 or Level 2. Although significant differences were found for the experiment cells at different problem-complexity levels, those differences are not indicated here because of the graphical complexity of the required presentation and because they are not important to the study of language effects.

Results for the average number of command lines are given in Figures 10, 11, and 12. These figures show that various language levels resulted in the least number of command lines for various problem-complexities. For problem-complexity level 1, for example, Language Levels 1 and 2 resulted in the least number of command lines (see Fig. 10). For problem-complexity level 2, Language Level 3 resulted in the least number of command lines. Finally, for problem-complexity level 3, Language Levels 1 and 3 resulted in the least number of command lines. Although the importance to problem solving of the number of command lines needed to solve a problem is not known, it is presumably an indicator of the efficiency of command-line use: fewer command lines suggest more commands per line, which may reflect differences in participants' abilities to build compound commands on a single command line.

Table 5

SUMMARY OF ANOVA RESULTS

PERFORMANCE MEASURE	LANGUAGE LEVEL	PROBLEM-COMPLEXITY LEVEL
NUMBER OF LINES	NS	.001
NUMBER OF LINES FOR TRANSIT	NS	.001
NUMBER OF LINES FOR HOMING	NS	.001
NUMBER OF "CR"	.001	.001
NUMBER OF "POINTS"	.001	.001
NUMBER OF KEYSTROKES PER LINE	.001	NS
NUMBER OF KEYSTROKES	.001	.001
NUMBER OF KEYSTROKES FOR TRANSIT	.001	.001
NUMBER OF STROKES FOR HOMING	NS	.001
TIME FOR TOTAL PROBLEM	NS	.001
TRANSIT TIME	NS	.001
TIME FOR HOMING	NS	.001
AVERAGE TIME PER KEYSTROKE	.001	NS

Analysis 2: Analysis of Variance

Purpose

To determine whether problem-complexity, language level, participant group, or any interaction among these factors, significantly affected the performance measures.

Method

Analysis of Variance.

Results

The analyses of variance for each performance measure are given in Appendix A. The results of the ANOVA's are summarized in Table 5.

The measures that, according to the ANOVA's, were significantly related to language effects were:

1. The average number of times the term "CR" was used to solve the total problem,
2. The average number of times the term "Points" was used to solve the total problem,
3. The average number of keystrokes per line to solve the total problem,
4. The average number of keystrokes to solve the total problem,
5. The average number of keystrokes to solve the transiting portion of the problem, and
6. The average time per keystroke to solve the total problem.

Note that the average time required to solve the total problem, as well as the average time to solve either the transiting portion or the homing portion, was not found to be significantly related to language level.

Problem-complexity had a significant effect on all variables except "the average number of keystrokes per line" and "the average time per keystroke." No significant differences were detected for the three groups of participants.

Table 4

Burr-Foster Q-test Values
For Language-Level Variances

Performance Measure	Q
1	.188
2	.239
3	.229
4	.177
5	.187
6	.236
7	.203
8	.248
9	.247
10	.191
11	.219
12	.209
13	.241

Q_c (P=3, df=29, a=.01) = .415

RESULTS

Analysis 1: Test of Variance

Purpose

Score variances for the three levels of problem-complexity and the three language levels were analyzed to determine if the variances were sufficiently homogeneous to permit use of the data directly, or if a transformation was required.

Method

The Burr-Foster Q-test to evaluate variance homogeneity.

Results

The Burr-Foster Q-test used the statistic

$$Q = \sum_{i=1}^{P} (S_i)^4 \Big/ \left(\sum_{i=1}^{P} S_i^2 \right)^2 \qquad (5)$$

where S_i is the level-i variance, and P is the number of levels for the problem at hand; in the experiments, for both problem-complexity and language level, P equaled 3. The Q table of Anderson & McLean (1974) provided the following critical values of Q for degrees of freedom (df) equal to 20 and 60:

$$Q_c = Q_{critical} \text{ (P=3, df=20, a=.01)} = .429$$

$$Q_c = Q_{critical} \text{ (P=3, df=60, a=.01)} = .367$$

A straight line interpolation between these values provided the critical value of Q for df=29 (30-1):

$$Q_c = Q_{critical} \text{ (P=3, df=29, a=.01)} = .415$$

Table 4 gives the Q values for each performance measure for the language-level data. As may be easily seen from the Table, the value of Q for each measure was less than the critical value of .415. Thus, there was no reason to believe that the variances were not homogeneous.

decomposed into a "transiting" first part, in which tests were requested to bring the number of effectiveness points close to the solution, and a "homing" second part, in which the exact solution was obtained. The homing part of the task involved some luck in finding the test that would result in the exact number of points needed. Hence, dividing the problem into the two parts effectively separated the performance during the interactive part of the problem, which was the part of interest, from the luck-dependent, terminal part. Measure scores were taken for the total task as well as for the two component parts of the task, transiting and homing. Data were collected for seven measures for the total task, and for six other measures, each adapted to one of the two parts of the task. The 13 measures were the:

1. Average number of command lines used to solve the total problem.
2. Average number of command lines to solve the transiting portion.
3. Average number of command lines to solve the homing portion.
4. Average number of times the term "CR" (or its implied form at Language Level 3) was used to solve the total problem.
5. Average number of times the term "Points" (or its implied form at Language Level 3) was used to solve the total problem.
6. Average number of keystrokes per line to solve the total problem.
7. Average number of keystrokes used to solve the total problem.
8. Average number of keystrokes to solve the transiting portion.
9. Average number of keystrokes to solve the homing portion.
10. Average time to solve the total problem
11. Average time to solve the transiting portion.
12. Average time to solve the homing portion.
13. Average time per keystroke to solve the total problem.

subject of a Freeze or Thaw command. The Language Level 3 abbreviations, which were all that level consisted of, are listed below, on the right. On the left are the unabbreviated commands.

Unabbreviated Commands	Language Level 3 Abbreviations
ADD _ _ POINTS TO CR _	A _ _ _
DELETE TEST _ _ _	D _ _ _
ADD MORE THAN _ _ POINTS TO CR _	M _ _ _
ADD LESS THAN _ _ POINTS TO CR _	L _ _ _
ADD _ _ POINTS TO SECTION _	A _ _ S _
ADD MORE THAN _ _ POINTS TO SECTION _	M _ _ S _
ADD LESS THAN _ _ POINTS TO SECTION _	L _ _ S _
FREEZE SECTION _	F _
THAW SECTION _	T _
AND	&
ERROR	***

Measures

A total of 13 measures were used to evaluate the performance of the participants. Before listing these measures, it is important to note that the task had to be divided into two parts. In the first part, the participant requested change-record (CR) tests, which resulted in assigning test effectiveness-points to each CR, as described in the section above on the experimental task. The participant could request tests to change the points for each CR, one CR at a time, or, at Language Levels 2 and 3, for a section of CR's. This first part of the task began with no points as yet assigned, which was the initial condition, and ended for each CR when the number of points assigned to that change-record was six points (or less) below the minimum allowed value, or was six points (or less) above the maximum allowed value. As soon as that condition came into being, it was possible to solve the problem with the addition (or deletion) of a single test. Consequently, at that point, the nature of the task changed from one of requesting tests that assigned points as efficiently as possible to one of discovering the exact test that would add (or delete) the exact number of points required to satisfy the problem's terminal conditions. Thus, the total problem could be

DELETE TEST _ _ _ .

AND (to form compound commands)

*** (error, delete this line)

Language Level 2

Language Level 2 permitted the commands of Language Level 1, but also permitted commands identifying groups of CR's, referred to as "Sections." There were seven such sections, each section identifying various CR's, as was shown above in Figure 7. Requesting X points for Section 1 was equivalent to requesting X points for CR1, CR2, and CR3. Likewise, requesting Y points for Section 7 was equivalent to requesting Y points for each of the 9 CR's, individually. Language Level 2 also permitted "ADD MORE THAN" or "ADD LESS THAN" commands, similar to those of the pretest language, but that could be applied either to a specified CR or to a section. The following is a complete listing of the commands available at Language Level 2:

ADD _ _ POINTS TO CR _ .

ADD _ _ POINTS TO SECTION _ .

ADD MORE THAN _ _ POINTS TO CR _ .

ADD LESS THAN _ _ POINTS TO CR _ .

ADD MORE THAN _ _ POINTS TO SECTION _ .

ADD LESS THAN _ _ POINTS TO SECTION _ .

DELETE TEST _ _ _ .

AND (to form compound commands)

*** (error, delete this line)

Language Level 3

Language Level 3 permitted all the commands of Language Level 2, but required the use of abbreviations, i.e., did not permit typing out the full command. In addition, the commands "FREEZE SECTION_" and "THAW SECTION_" were available. "FREEZE SECTION_" prohibited any addition or deletion of tests that had non-zero effectiveness points for CR's in the section specified while "THAW SECTION_", entered after a companion Freeze command, would again permit addition, or deletion, of tests affecting CR's in that section. Note that only a section, but not a CR, could be the

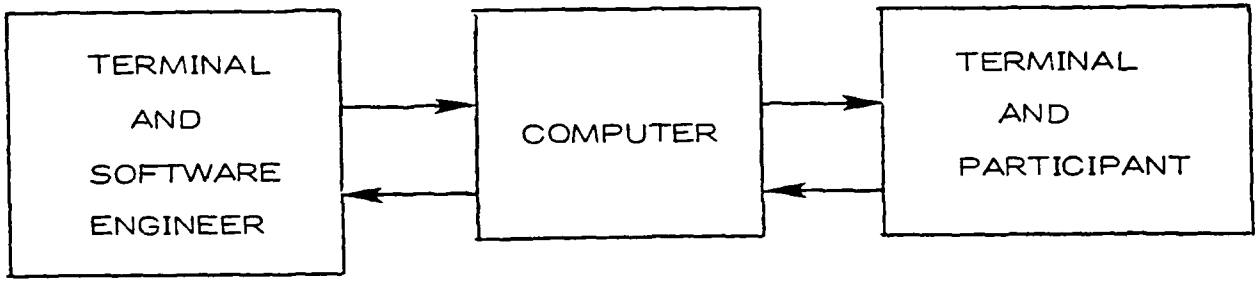


Figure 9. Communication System for the Experiments

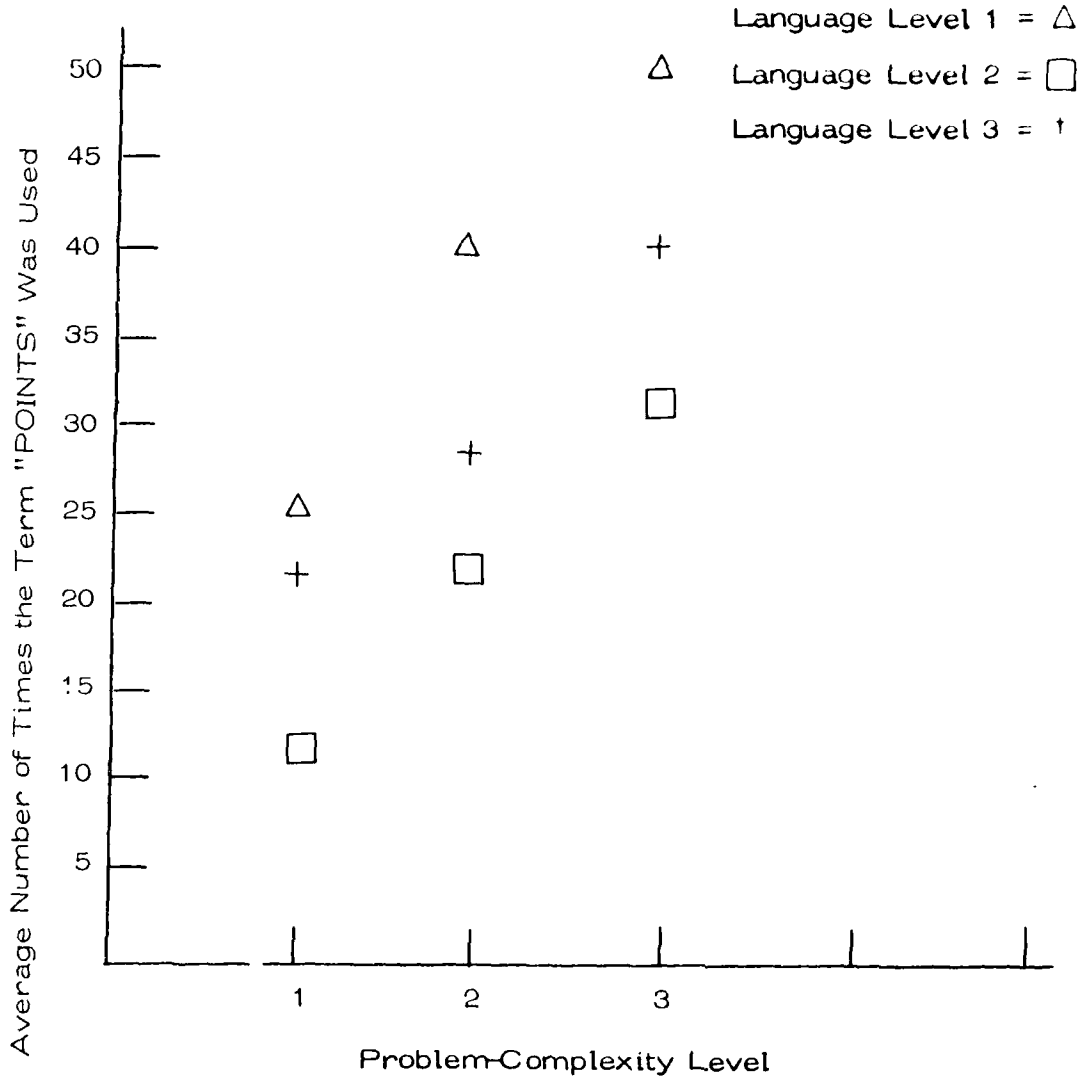


Figure 14. Average Number of Times the Term "POINTS" Was Used To Solve the Total Problem

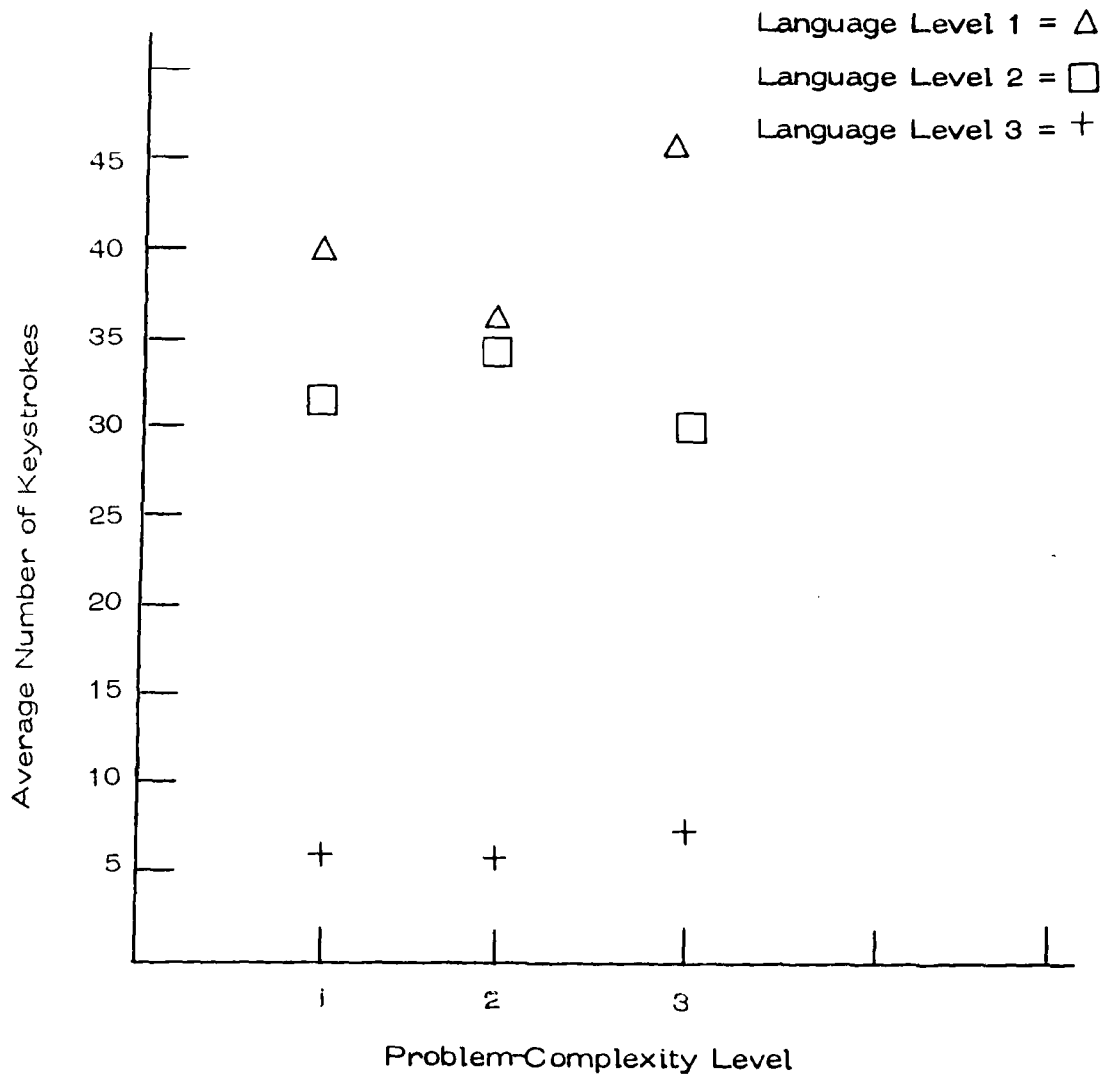


Figure 15. Average Number of Keystrokes Per Line to Solve the Total Problem

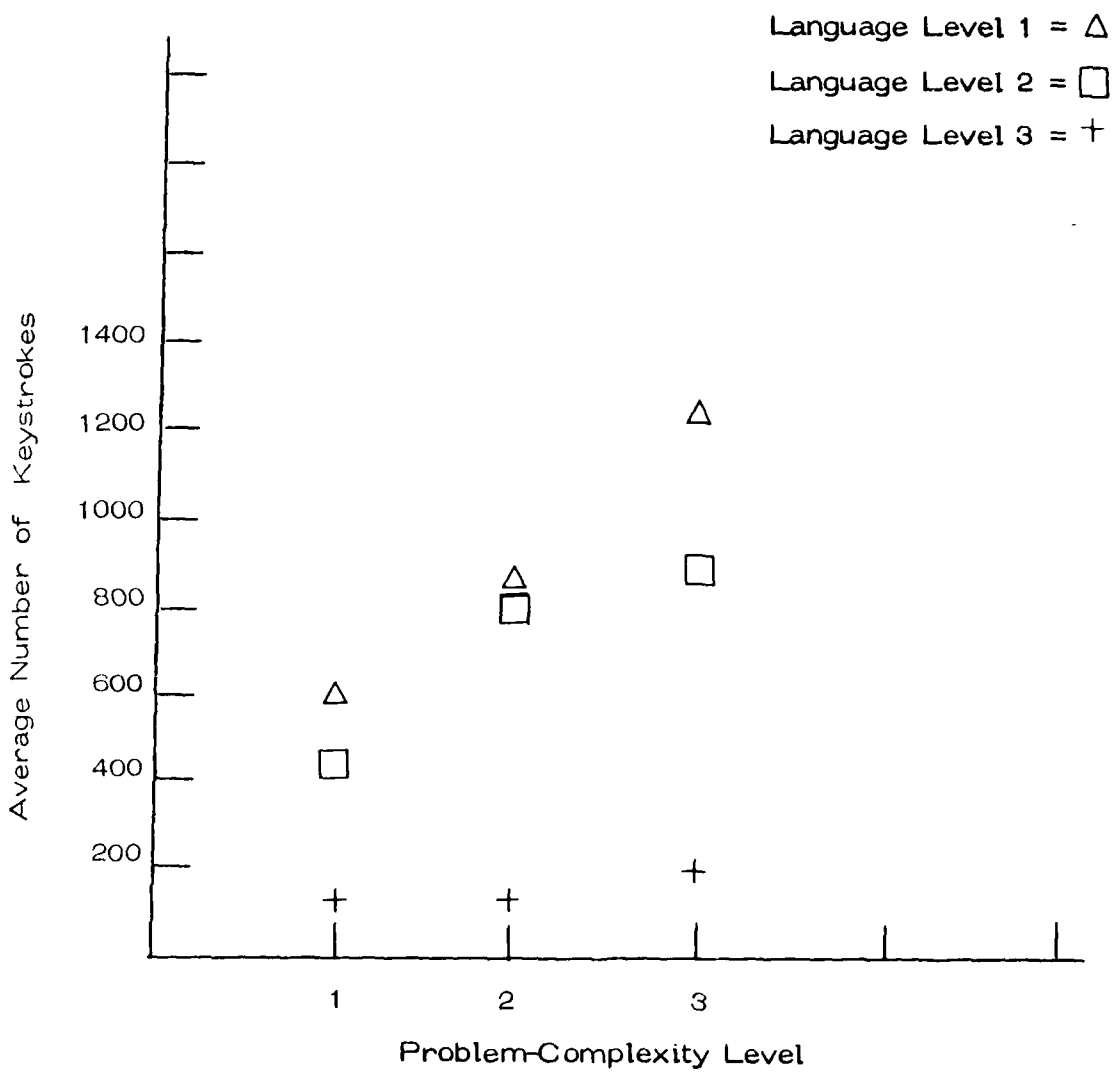


Figure 16. Average Number of Keystrokes To Solve the Total Problem

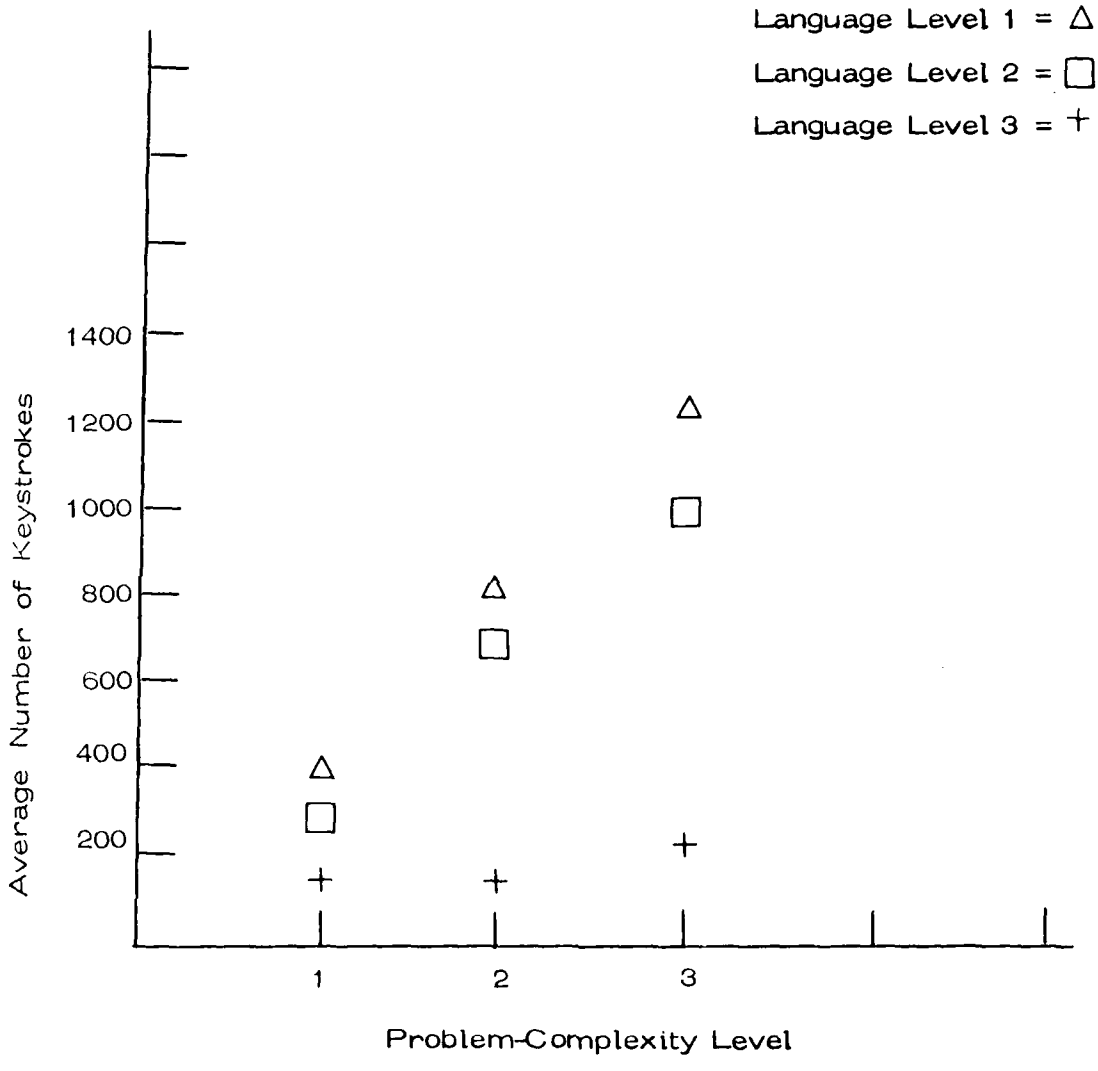


Figure 17. Average Number of Keystrokes To Solve the Transiting Portion

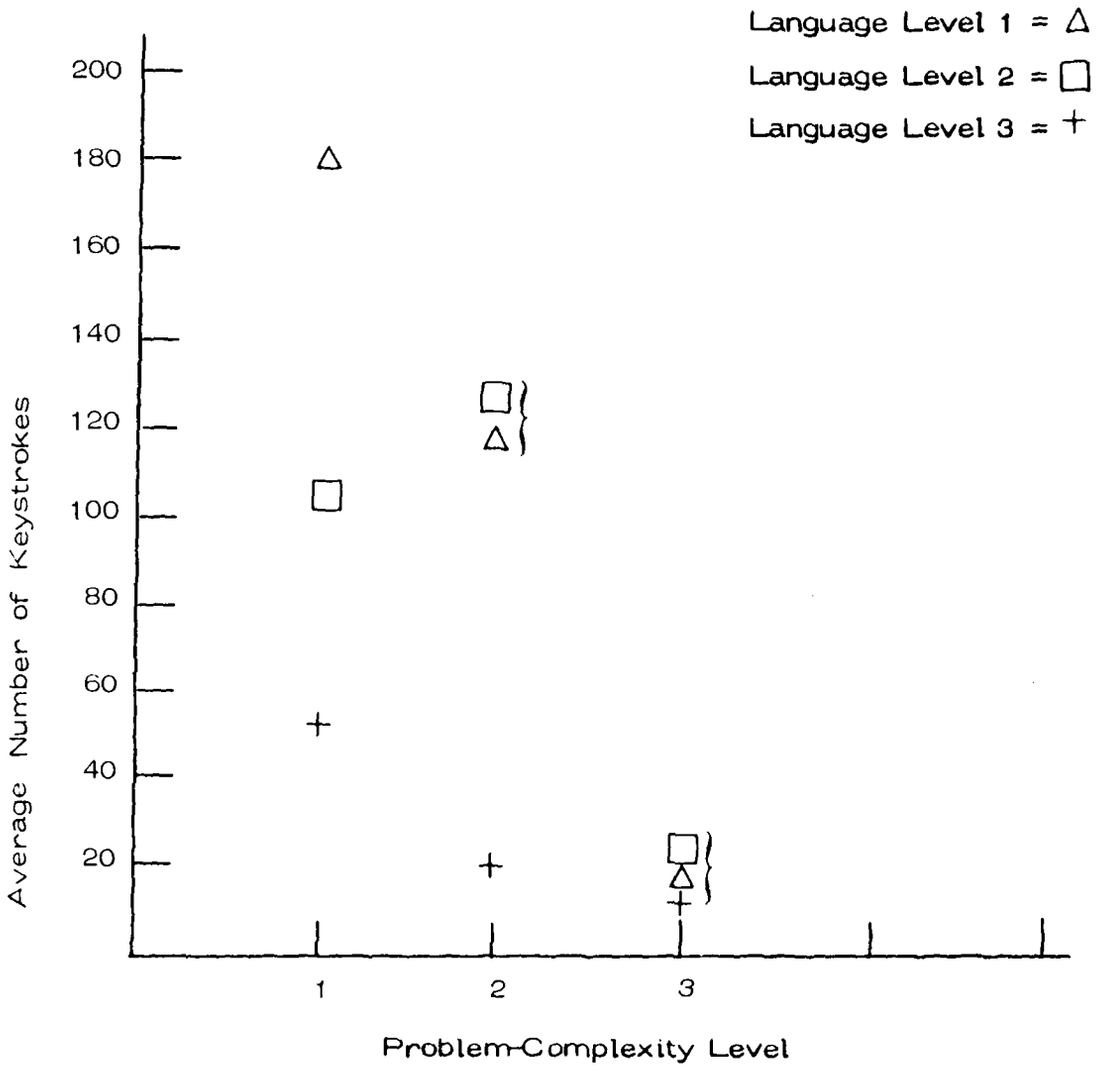


Figure 16. Average Number of Keystrokes To Solve the Homing Portion

had been required at Language Level 1, where each CR had to be treated individually. The abbreviations introduced at Language Level 3 permitted yet a further reduction in the number of keystrokes required to solve the total problem, as well as to solve the transiting and the homing portions of the problem.

Results for the measures "average time to solve the total problem", "... to solve the transiting portion" and "... to solve the homing portion" are given in Figures 19, 20 and 21, respectively. Although it might have been expected that the average time to solve the homing portion would be independent of problem-complexity, as shown in Figure 21 the time to solve the homing portion apparently decreased with increasing problem-complexity! Further investigation revealed that, because the time to solve the total problem was limited to a one hour period, as increased time was needed to solve the transiting portion of the problem, less time, on the average, became available to solve the homing portion. Consequently, although not all participants were limited by the problem's time constraint, some were limited, and that biased the average result. But in spite of this occasional encroachment of the transiting time onto the homing time, the relative effects of the language levels on the homing time could still be inferred. There were none. The data presented in the figures do not reveal any consistent trend indicating the superiority of one language level over another for the time to solve the total problem or for the time to solve the transiting portion; consequently, the homing portion was similarly unaffected.

The results for the average time per keystroke, shown in Figure 22, clearly indicate that, at Language Level 3, where abbreviations were used, the time per keystroke was 4 to 5 times greater than that at Language Levels 2 and 3. This increase in the time per keystroke, when using abbreviations, suggests that the use of abbreviations may not in itself decrease the time required to solve a problem, or even the time to enter a command, by an amount proportional to the reduction in the number of keystrokes allowed by the abbreviations.

Analysis 4: The Effect of the Number of Commands in a Language and the Number of Keystrokes for a Command on the Average Time to Enter a Command

Purpose

The previous analyses considered the effect of the three specific language-level designs on the time to enter a command. However, there was

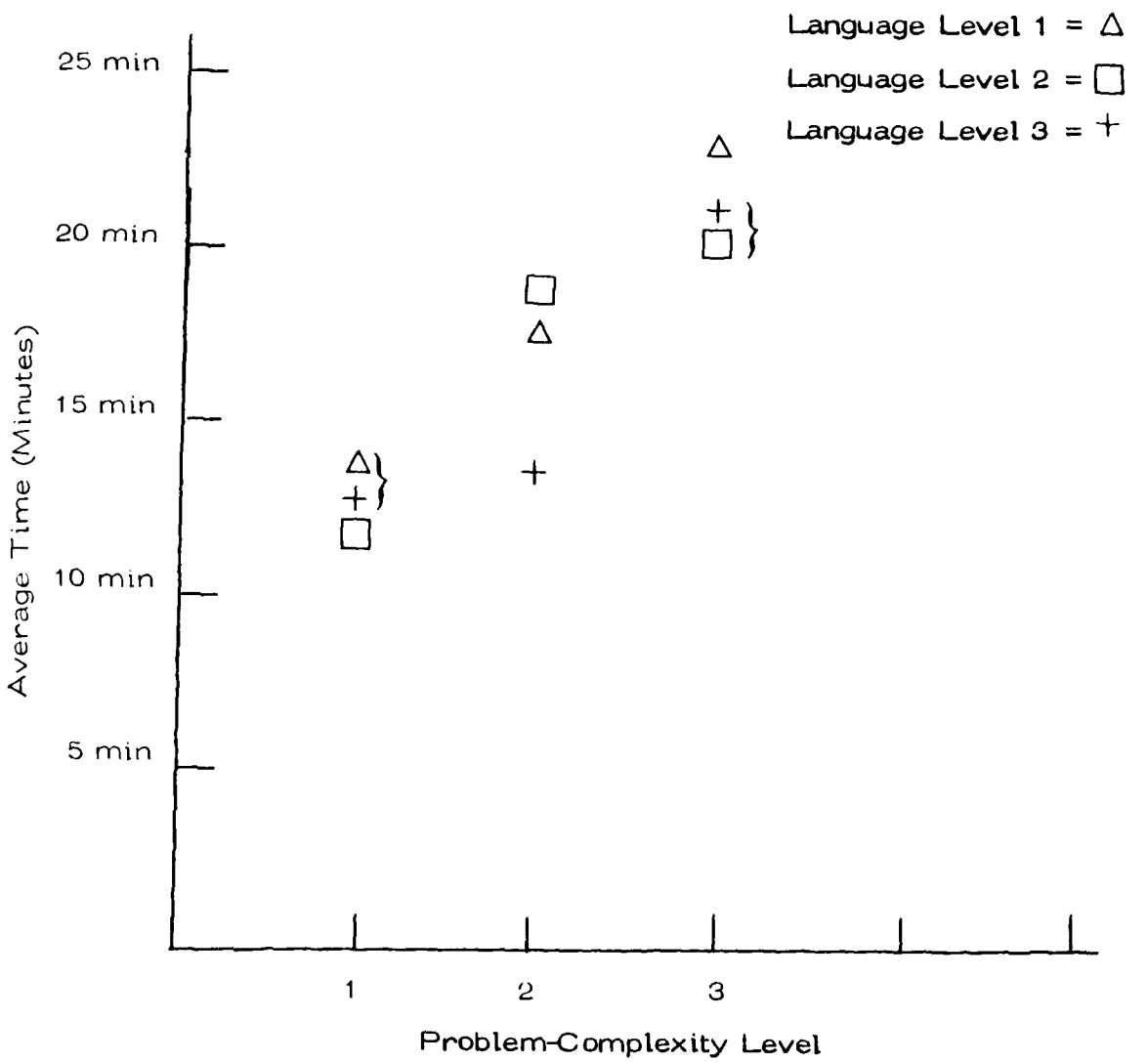


Figure 19. Average Time to Solve the Total Problem

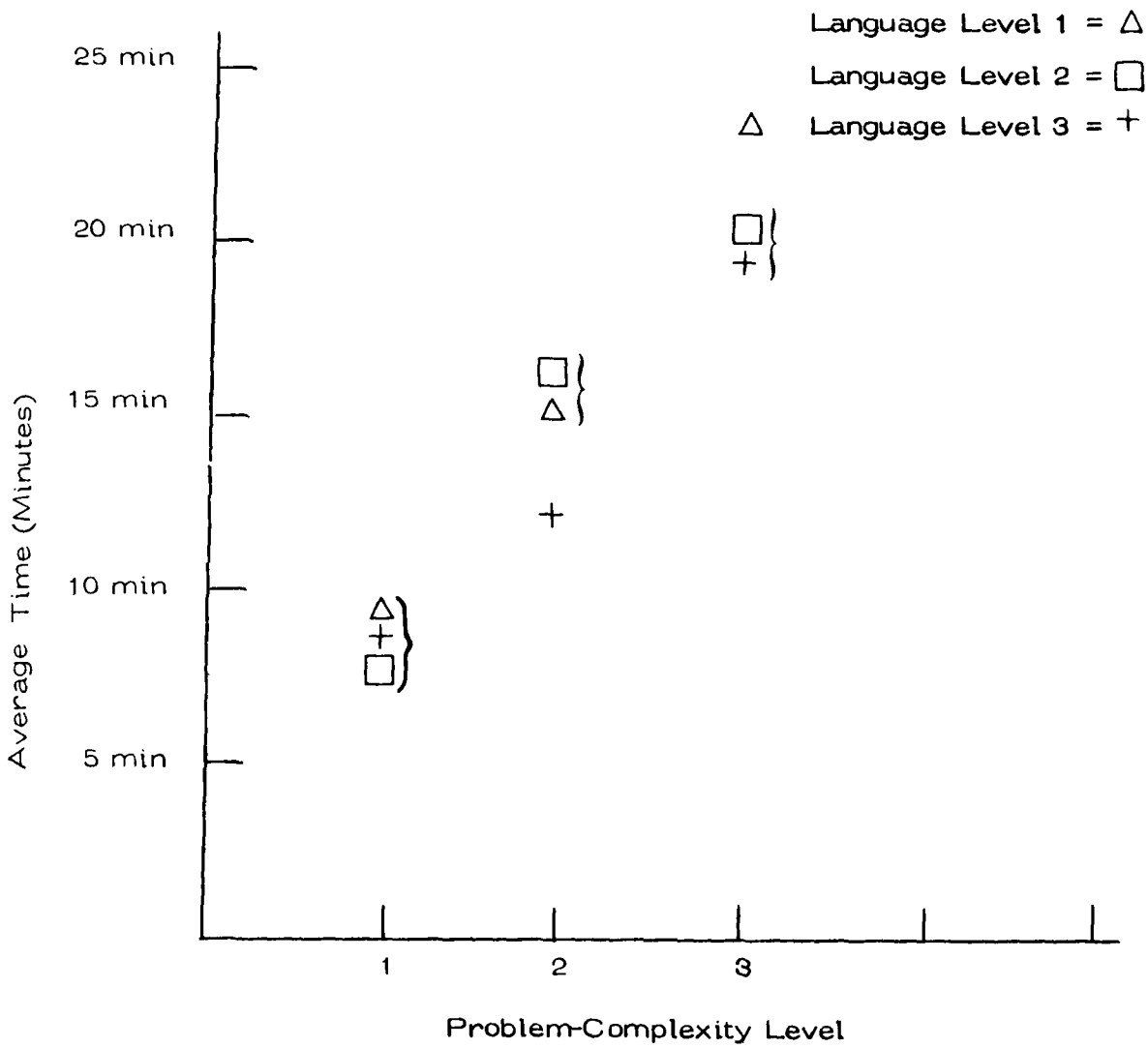


Figure 20. Average Time to Solve the Transiting Portion

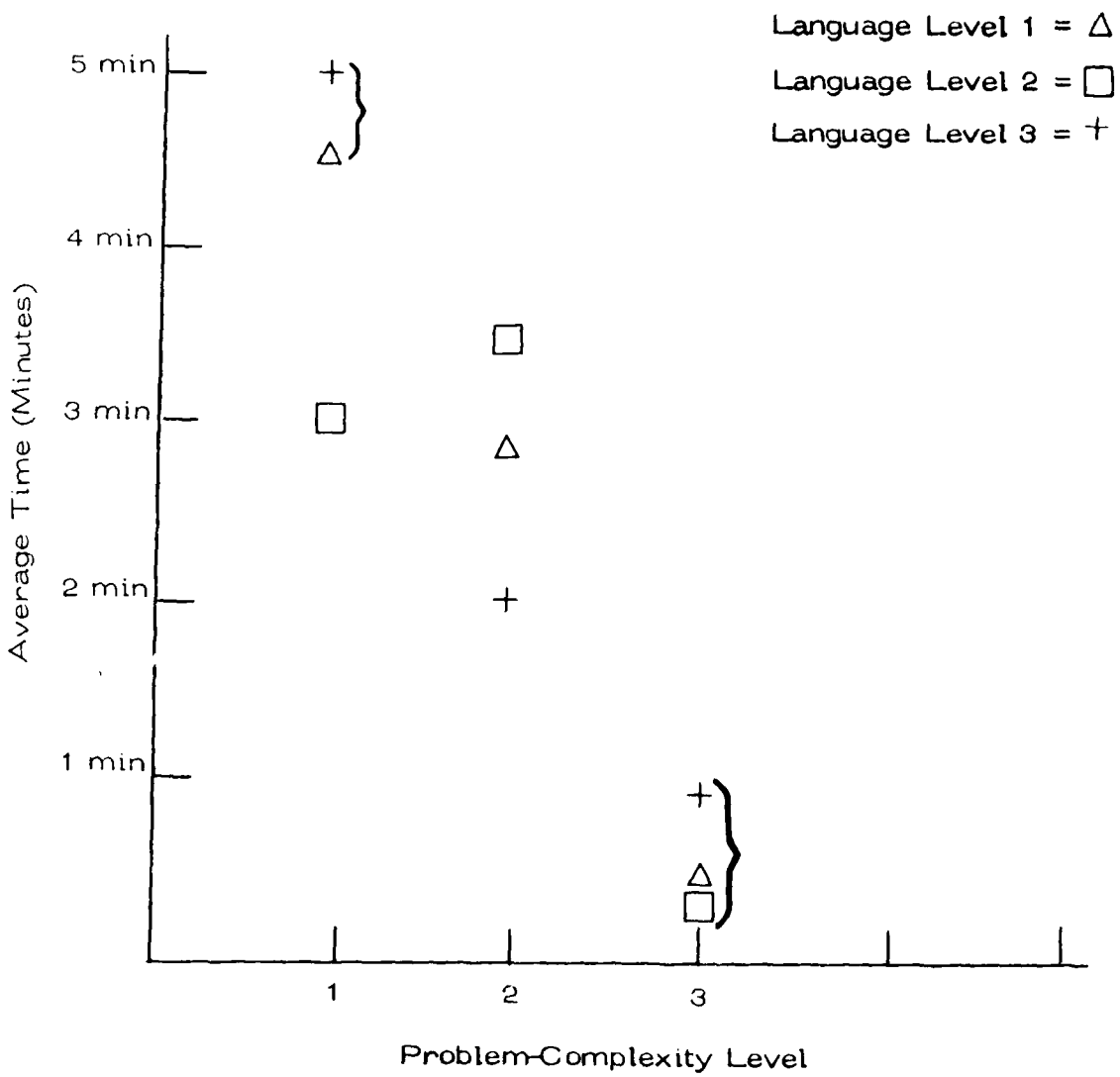


Figure 21. Average Time to Solve the Homing Portion

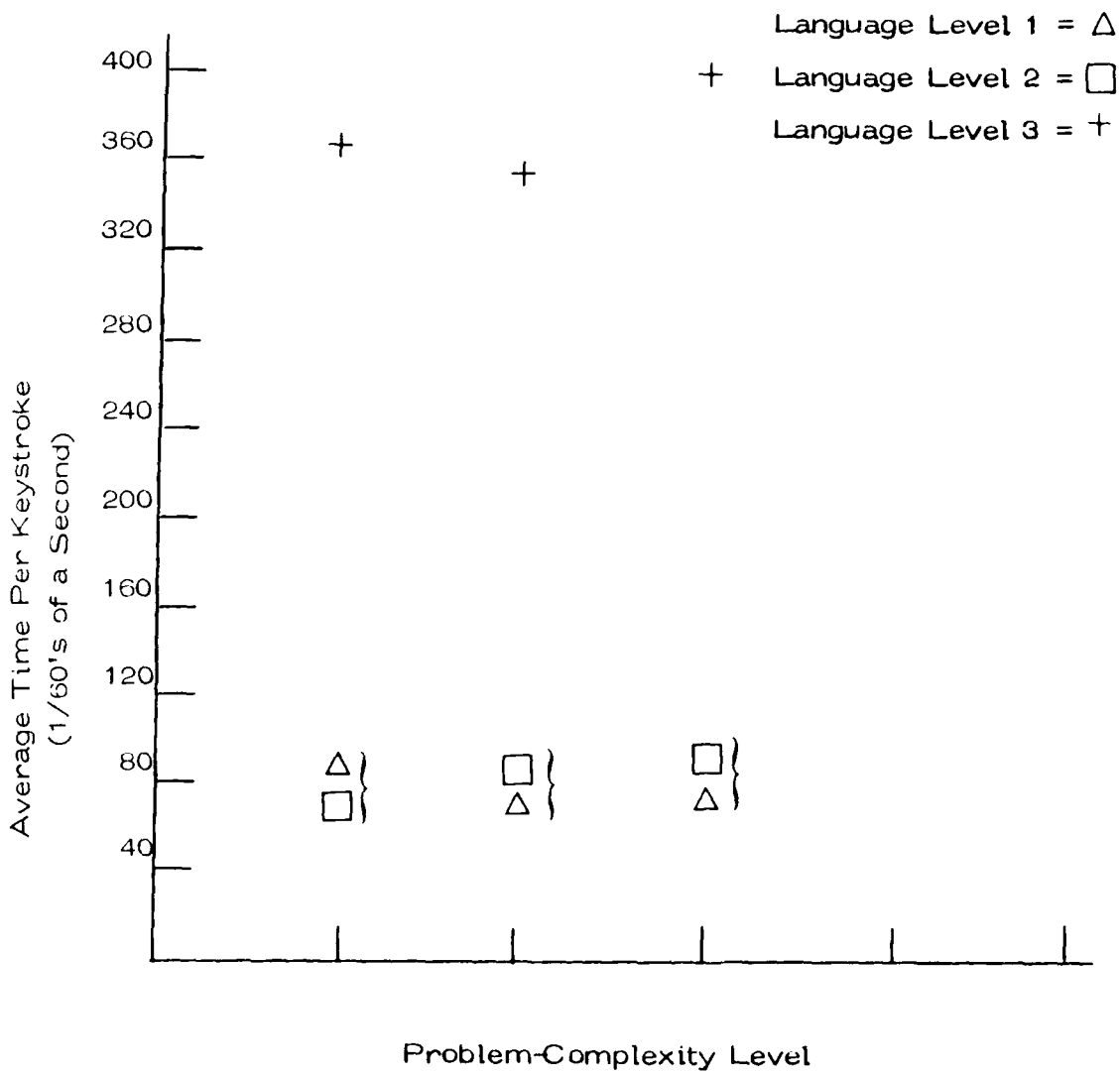


Figure 22. Average Time Per Keystroke To Solve the Total Problem

evidence to suggest that the number of keystrokes required to enter a command and the number of commands in a language may have been among the underlying factors influencing the time required to enter a command. The purpose of this analysis was to determine whether these factors, or the frequency of usage of commands and/or other factors,were significant predictors of the time required to enter a command.

Method

Step-wise univariate and multivariate regression analyses were performed with the average time to enter a command as the dependent variable. The independent variables investigated were:

1. First command on line,

2. Number of characters (i.e., keystrokes) in a command, and

3. Number of commands in a language.

In a separate analysis, the following independent variable was used:

4. Number of times a command was entered (i.e., command frequency).

Several sets of data were used for the dependent variable. They were the:

1. Average time to enter all commands,

2. Average time to enter commands that were not first on a line,

3. Average time to enter commands that were first on a line, and

4. Average time to enter commands not first on a line, but excluding the error command.

Results

Table 6 gives the number of keystrokes required to enter a command or command element. Where the number of keystrokes was a function of the number of digits to be entered, which could require from 1 to 3 keystrokes, two digits were assumed.

Type Number	Command SYNTAX	Number of Keystrokes	Used in Language: Level			
			Pretest	1	2	3
1	ADDΔ	4	×	×	×	
	A	1				×
2	DELETEΔTESTΔXXX+	14, 15, or 16 depending on one, two, or three digits (XXX)	×	×	×	
	DXXX+	3,4,5 depending on one, two, or three digits (XXX)				×
3	ADDΔMOREΔTHANΔ	14	×		×	
	M	1				×
4	ADDΔLESSΔTHANΔ	14	×		×	
	L	1				×
5	TOΔCRΔX+	8	×	×	×	
	X+	2				×
6	XXΔPOINTSΔ	10	×	×	×	
	XX	2				×

Table 6. Number of Keystrokes for Each Command or Command Element

Type Number	Command SYNTAX	Number of Keystrokes	Used in Language: Level			
			Pretest	1	2	3
7	TOΔSECTIONΔX+	13			X	
	SX+	3				X
8	ANDΔ	4	X	X	X	
	(SHIFT)&	2				X
9	*** (error)	3	X	X	X	X
10	FREEZE SECTION[1] X					
	FX+	3				X
11	THAW SECTION[1] X					
	TX+	3				X
	NUMBER OF COMMANDS		6	6	9	11

Δ = Blank
\+ = Blank or Return
X = Digit (0-9)

Note 1- Command element available only in abbreviated form

Table 6 (Continued). Number of Keystrokes for Each Command or Command Element

Table 7 gives the correlations among the independent variables
ıd the correlation of each variable with the average time to enter a
ɔmmand, for all commands (data set 1).

Table 8 gives the results of the step-wise regression analysis
ır data set 1. The only variable significantly related to the average
me to enter a command, for all commands, was "First variable on a
ne". This variable explained 80% of the variance of the average time
ɔ enter a command.

Table 9 gives the correlations among the independent variables
ıd the correlation of each variable with the average time to enter a
ɔmmand, for commands not first on a line (data set 2).

Table 10 gives the results of the step-wise multivariate regression
nalysis for the dependent variable "all commands not first on a line".
ı this case, two independent variables, the "number of characters in the
ommand" and the "number of commands in a language" were significant
ı explaining 74.5% of the variance of the time to enter a command not
.rst on a line.

A step-wise multivariate regression analysis for data set 3,
/hich included only those commands that were first on a line, using the
ndependent variables listed above resulted in no significant correlations
/ith the "average time to enter a command".

Table 11 gives the correlations among the independent variables
nd the correlation of each variable with the average time to enter a command
ot first on a line, but exclusive of the error command (data set 4).

Table 12 gives the results of the multivariate regression for
ata-set 4. The "number of characters in a command" and the "number of
ommands in a language" and the "number of times a command is used",
'ere all significant predictors, together explaining 81% of the variance.

A plot of the average time to enter a command or command
lement for commands not first on a line is shown in Figure 23. The lines
onnecting the plotting symbols are used to connect those symbols for
larity of presentation; the lines themselves have no meaning. Of special
ıterest was the trend for all the commands and command elements available
t Language Level 1 to have an increased entry time when they were used at
.anguage Level 2. Yet the only difference between these language levels was
ıe addition of the new command elements "Section," "Add more than",
nd "Add less than". This evidence suggests that the number of commands
nd/or the nature of the additional commands (or command elements) can
ıfluence entry time for all commands in a language.

rategy selections, which were assumed to represent the most effective rategies as perceived by the participants, work inequalities implied those strategy selections were established. The argument was that, nce both commands (the set of CR commands or the one Section Command) hieved the same result, any preference for one command over another .d to be due to a difference in the perceived amount of work required to .e the one command over the other.

esults

Table 13 shows that, for Language Levels 2 and 3, the frequency use of Section commands decreased as the participants progressed om the 1st through the 2nd and 3rd stages of the experimental problem. his is the result that was expected from the nature of the problem design.

However, a difference occured in the use of CR commands ersus the use of Section commands between these same two language evels. Table 14 shows that, at Language Level 2, a total of 354 CR ommands were used, but that, at Language Level 3, 527 CR commands were sed. The Section commands were reversed in frequency, with Language .evel 2 showing a greater usage of Section commands than Language Level (337 vs. 307). The same total task was accomplished, but in different ways.

The cognitive steps required to use each language level are hown in: Figure 27 for Language Level 1, Figure 28 for Language Levels and 3 for CR commands, and in Figure 29 for Levels 2 and 3 for Section ommands. In order to use Section commands at Language Levels 2 and 3, he participant had to analyse and then commit to memory the effectiveness- oint adjustments required for multiple (3 to 9) CR's prior to inputing each Section command. In contrast, when CR commands were used at any of he levels, each CR was analysed and the associated command was input ne CR at a time. A comparative (qualitative) summary of the relative ental work is shown in Figure 30. The terms defined in that figure were sed to construct a mental "work-area," which, together with the keystroke odel, explained the strategy selections of the participants.

Figure 31 shows the results of the keystroke analysis. At anguage Level 1, 22 keystrokes were required to add points to a single R, the same number of keystrokes as at Language Level 2, when he CR command was used. However, at Language Level 2, when the Section command was used, 27 keystrokes were required. Since hose 27 keystrokes could request effectiveness points for 3, 6, or CR's, the number of keystrokes per CR ranged from 9 to 3. This gain

reference to many (from three, to as many as nine) CR's simultaneously. As work on the problem progressed, i.e., as the effectiveness points for one or more of the CR's approached the required tolerance limits, a different strategy, that of employing individual CR commands, was thought to gain in importance. This sequential CR strategy was thought to provide the control necessary to obtain tests with the correct number of effectiveness points for each CR, so that the total number of effectiveness points for each CR satisfied the pre-specified tolerance limits. To test these suppositions, the problem was divided into thirds, according to the cumulative number of keystrokes, and the command types used during each third were tabulated.

The second part of the analysis was motivated by a result from the first part, where it was found that an unexpected change in participant strategies occured between Language Levels 2 and 3. In particular, it was noted that, at Language Level 2, there was a tendency, as expected, to use the powerful Section commands; at Language Level 3, however, there was a tendency to use the less powerful, individual CR commands. Apparently, participants tended to perceive that Section commands were more effective at Language Level 2, but that CR commands were more effective at Language Level 3. The question that arose was: Why did a change occur in the preferred command type when both language levels offered both CR and Section commands?

It was reasoned that the participants' strategy-choices might perhaps be explained by keystroke effectiveness. Alternatively, their choices might be explained by a variation in the perceived total amount of work involved (mental work required to use a command plus manual work required to type in the command) versus the benefits obtained. Both keystroke-effectiveness and total-work models were developed in order to discover possible reasons for the observed difference in strategy.

The keystroke-effectiveness model was simply based upon the number of CR's that could be referenced per keystroke.

The total-work model was constructed by first analysing the steps required to use CR and Section commands at each language level. In order, however, to avoid having to analyse differences in perceived benefits for various commands each producing different results, a common task was defined by constructing a combined set of CR commands that together accomplished the same result as a single Section command. (The common task was to request 9 points for each of three specified CR's). The comparative work required to use either the set of commands or the Section command was then identified. Finally, based on the observed

Thus, the first keystroke in Figure 25 is an "A", the first character on the line, which required a 14.7 second inter-stroke time. The key-strokes "D" and "D" are plotted next, in columns 5 and 3, respectively, to indicate inter-keystroke times of .5 and .3 seconds, respectively. The completed "ADD" command element is followed next by a space (Δ) finally by a "9". The total command sequence is "ADD 9 POINTS TO CR 2 and ADD 0 POINTS ...". Note, in this plot, which is of a typical command sequence at Language Levels 1 and 2, that the first character on the line required a long entry time. Subsequently, command characters required a shorter entry time, ranging from .3 to .7 seconds. The digits (9, 2, 9), however, also required a long entry time, ranging from 1 to 2.9 seconds.

In contrast to the typical command pattern at Language Levels 1 and 2 was the typical command pattern at Language Level 3, shown in Figure 26. In the first example, the first command element was the abbreviation "A" for the command element "ADD." The abbreviated element required 17.3 seconds. The subsequent number 9 required 8.1 seconds, and the Section command element "S" required 11.7 seconds. All the command characters now required considerable more time to enter than at Language Levels 1 and 2. This suggests that a considerably length of time was required to think of an abbreviation before it could be used. Additional analyses, to be described subsequently, were undertaken to evaluate this supposition. Finally, note how the time to input a specific abbreviation decreased with repeated, sequential use: thus, entry of the Section command element "S" required 11.7, 4.1, 1.0 and,finally, .7 seconds.

Analysis 5

Purpose

This analysis investigated the strategy employed by participants in their selection of commands, as a function both of language level and of the degree of completion of the problem.

Method

A different strategy for selecting different command types was thought to be appropriate at successive stages of the problem. At the beginning, the problem was thought to require efficiency in requesting tests that provided effectiveness points for each CR. Thus, use of the powerful Section commands at Language Levels 2 and 3 was thought to be particularly appropriate, because these commands permitted

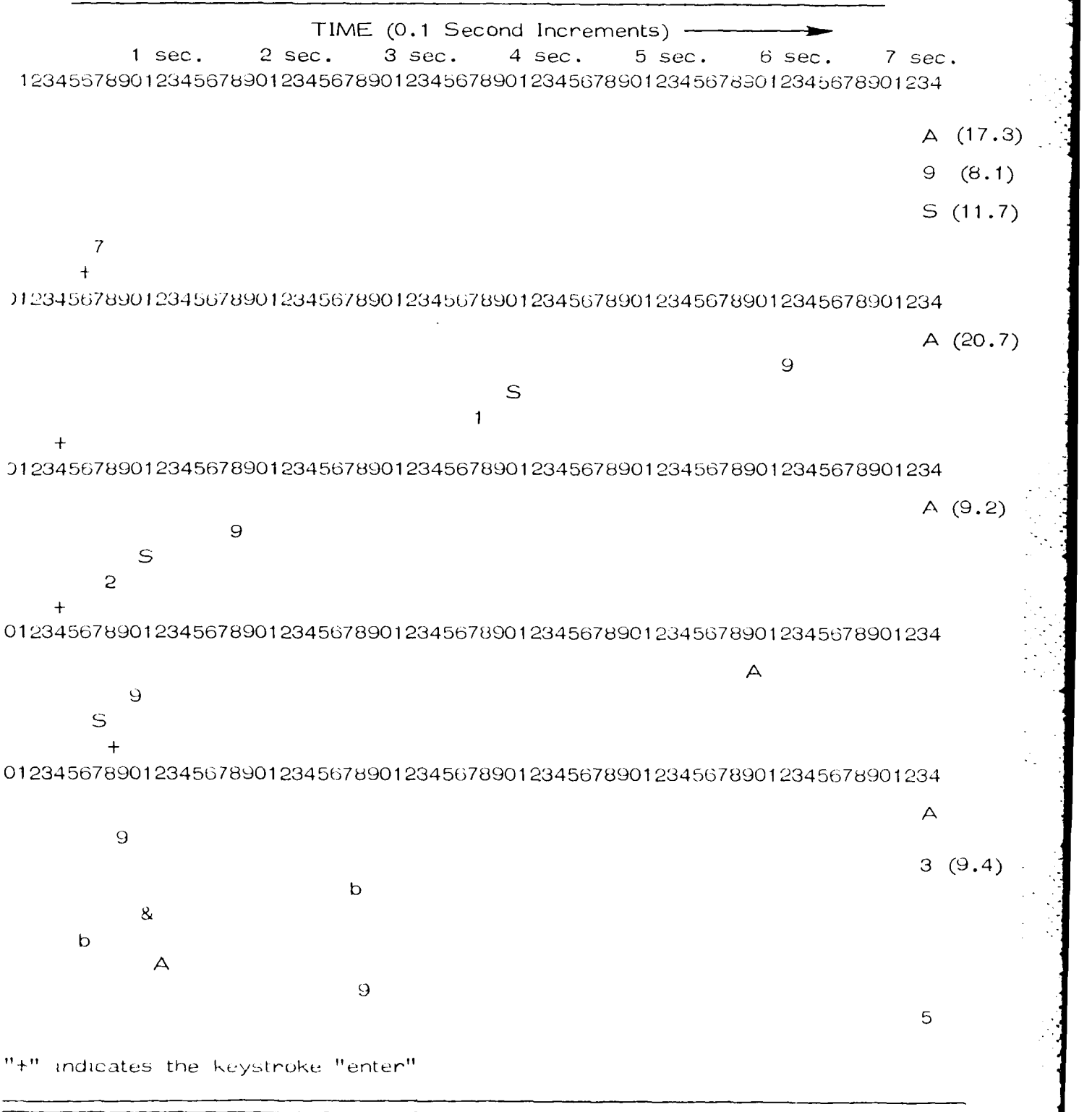


Figure 26. Time/Characters-Sequence Plot for Representative Commands at Language Level 3

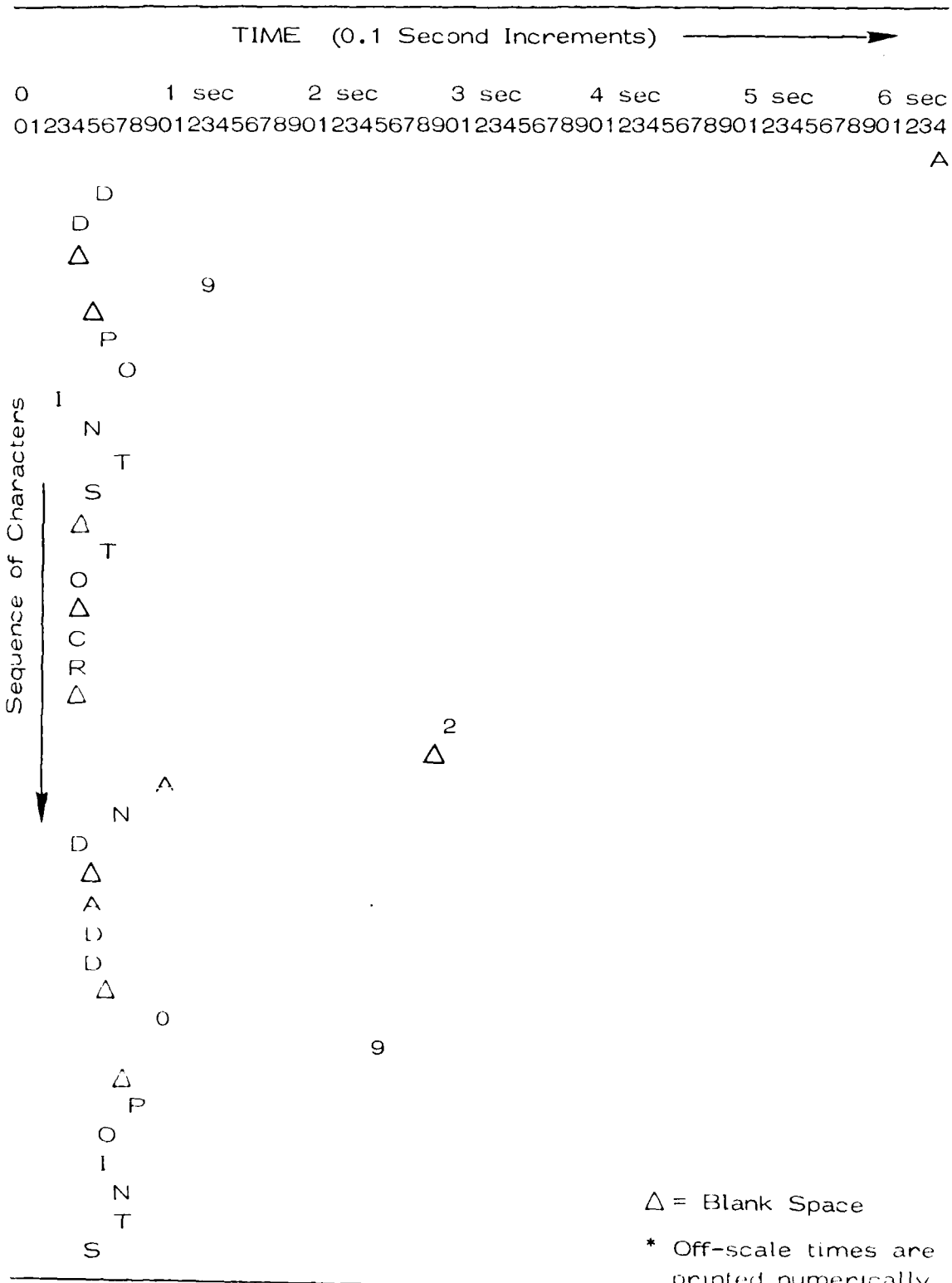


Figure 25: Time/Character-Sequence Plot for the Language Level 1 Sequence "ADDΔ9ΔPOINTSΔTOΔCR2ΔANDΔADDΔ09ΔPOINTS"

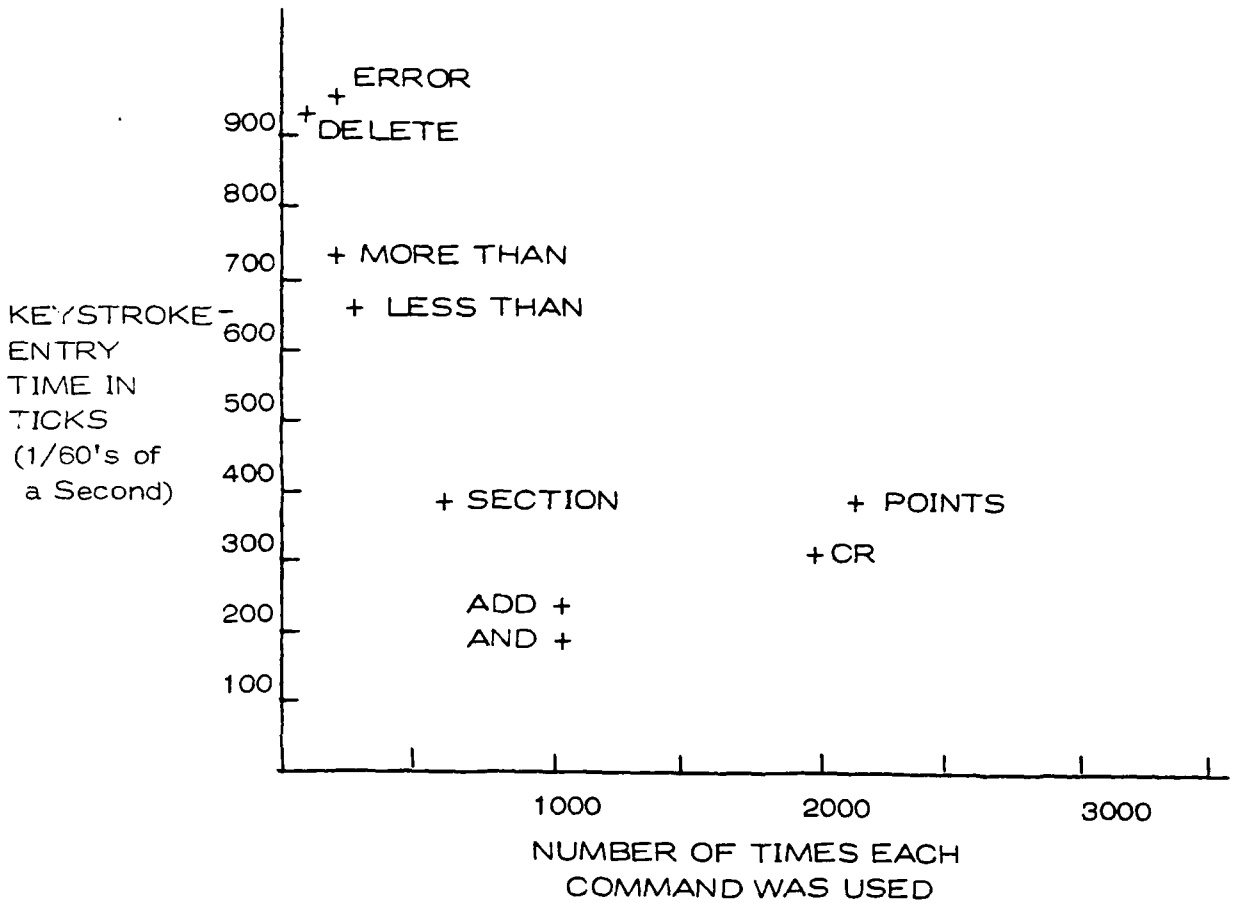


Figure 24. Average Entry Time vs. Command Usage

Another trend suggested by the plot was that of an increase in entry time for short commands (or command elements), such as "ADD" when abbreviations were introduced (at Language Level 3) even though the entry time for medium-length commands, 8 to 12 characters in length, remained almost constant. Only the longer commands, from 14 to 16 characters in length, realized a reduction in entry time with abbreviations.

The reader may want to refer again to Table 6 for the number of keystrokes in a command. When using abbreviations the command entry times, except for the "error" and "AND" commands, seemed to collect in the 220 to 360 tick (1 tick = 1/60 of a second; 220-360 ticks = 3.66-6.00 seconds) time range. This range, however, is long for the time to push a single key, which suggests that the interval between keystrokes was dominated by a problem-solution or problem-strategy development time and was not simply a keying time.

A final observation suggested by the plot is that the entry times for the "error" and "AND" commands were influenced in a different way from the other commands by the language-level designs. Apparently, the time to detect an error and key in an error command increased almost 100% at Language Level 2 compared to Language Level 1, suggesting that one penalty for adding more powerful commands was an increased time to detect and correct errors in all command sequences. Further, since the error entry-time at Language Level 3 did not decrease to fit into the range of the other abbreviated commands, one suspects that the mental processes used by participants to detect errors differed from those used to structure problem-solving commands.

The "AND" command required approximately the same short entry time, independent of the language level used. Apparently, a connective is not greatly influenced by the use of abbreviations, by language design, or by problem-solution time.

The average entry time as a function of command usage is shown in Figure 24. This figure clearly suggests that an inverse relationship existed between command usage and entry time.

Plots of representative character sequences vs. keystroke-entry time for Language Levels 1 and 3 are given in Figures 25 and 26, respectively. These plots show the characters in a representative command-sequence vertically, starting, from the top and reading down. The time between keystrokes (interstroke time) is shown on the ordinate by numbers indication 0.1 second increments. Keystroke intervals requiring large off-the-scale times are plotted at the right edge, with the time in parentheses.

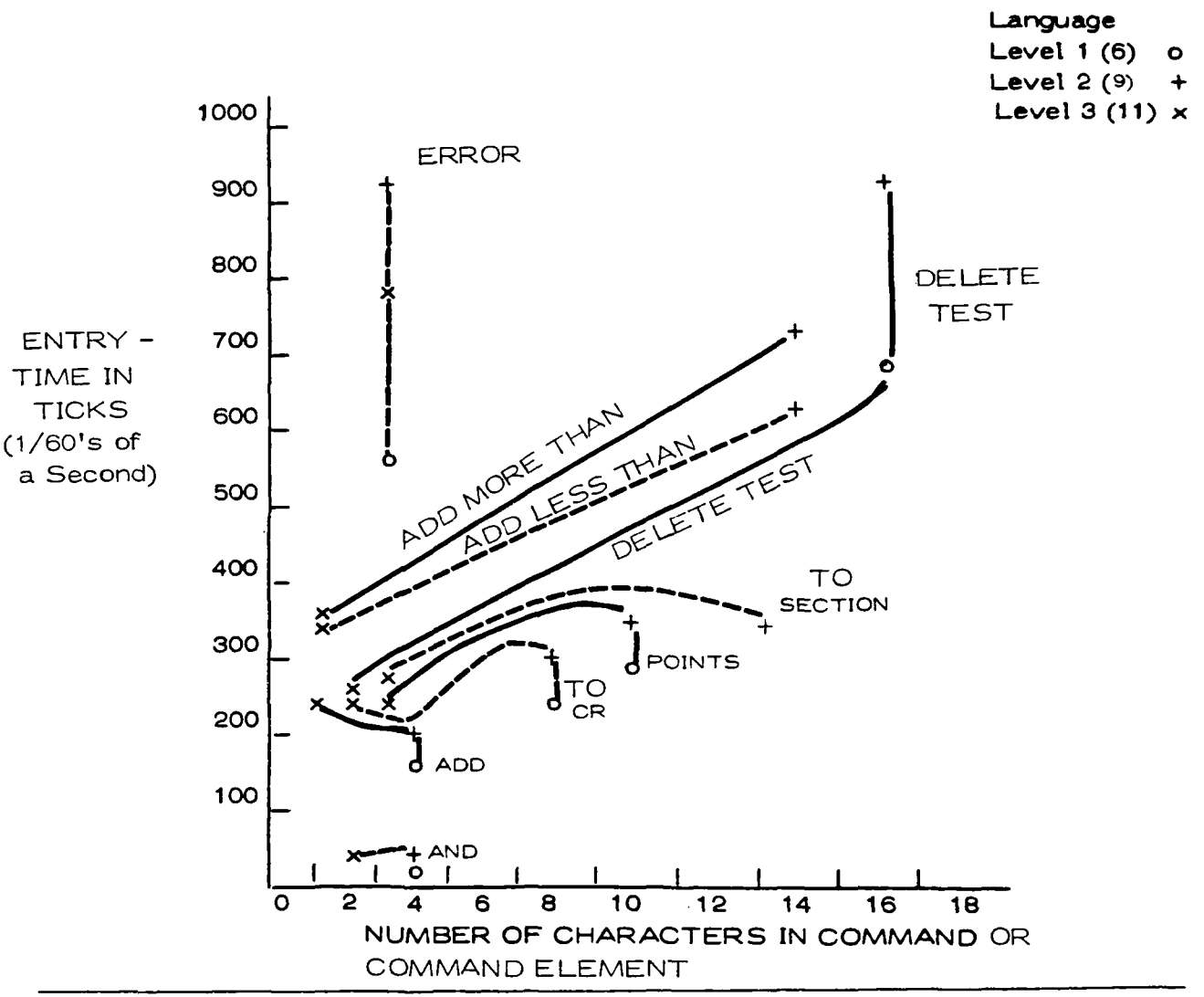


Figure 23. Command or Command Element Length vs. Average Time to Enter a Command or Command Element (For Commands Not First on a Line)

Table 12

Multivariate Regression For
All Commands, Except the Error Command, Not First On a Line

Dependent Variable: Average Time to Enter A Command

Independent Variable	Coefficient	Variance Explained	t-Value	P≤
Number of Characters in a Command	33.7		6.59	.001
Number of Commands in a Language	31.3	81.8 %	2.50	.01
Number of Times a Command is Used	-0.07		-2.60	.01

Table 11

Correlation: Average Time to Enter a Command
vs.Various System Variables

For all Commands Not First On a Line,
Except the Error Command

		Correlations			
		Among Variables			Of Each Variable With the Average Time
Index	Variable	1	2	3	
1	Number of Characters in a Command		-.493	-.247	.784
2	Number of Commands in a Language			-.187	-.072
3	Command Frequency				-.565

Table 10

Multivariate Regression For
All Commands Not First On a Line

Dependent Variable: Average Time To Enter Command

Independent Variable	Regression Coefficient	Variance Explained	t-Value	P ≤
Number of Characters in a Command	39.0	74.5%	7.231	.001
Number of Commands in a Command	44.3		3.037	.01

Table 9

Correlation: Average Time to Enter a Command
vs. Various System Variables

For All Commands
Not First on a Line

Index	Variable	Correlations Among Variables 1	2	3	Of Each Variable With the Average Time
1	Number of Characters in a Command		-.426	-.122	.472
2	Number of Commands in a Language			-.132	-.030
3	Command Frequency				-.639

Table 8

Univariate Regression For All Commands
Dependent Variable: Average Time toEnter Command

Independent Variable	Regression Coefficient	Variance Explained	t-Value	P≤
1st Command on a Line	1561.8	80%	12.15	.001

Table 7

Correlation: Average Time to Enter a Command
vs. Various System Variables

For All Commands

Index	Variable	Correlations Among Variables 1	2	3	4	Of Each Variable with the Average Time
1	1st Command on a Line		.043	.017	-.144	.894
2	Number of Characters in a Command			-.414	-.121	.200
3	Number of Commands in a Language				-.116	-.020
4	Command Frequency					-.292

Table 13

Frequency of "Section" Commands vs. Problem Stage

	Language Level	Frequency
1st Third of Problem	L_1	Section Command Not Available
	L_2	185
	L_3	168
2nd Third of Problem	L_1	Section Command Not Available
	L_2	90
	L_3	91
3rd Third of Problem	L_1	Section Command Not Available
	L_2	62
	L_3	48
TOTALS FOR PROBLEM		
	L_1	--
	L_2	337
	L_3	307

Table 14

Frequency of CR Commands vs. Problem Stage

	Language Level	Frequency
1st Third of Problem	L_1	453
	L_2	66
	L_3	190
2nd Third of Problem	L_1	391
	L_2	158
	L_3	163
3rd Third of Problem	L_1	306
	L_2	130
	L_3	174
TOTALS FOR PROBLEM		
	L_1	1150
	L_2	354
	L_3	527

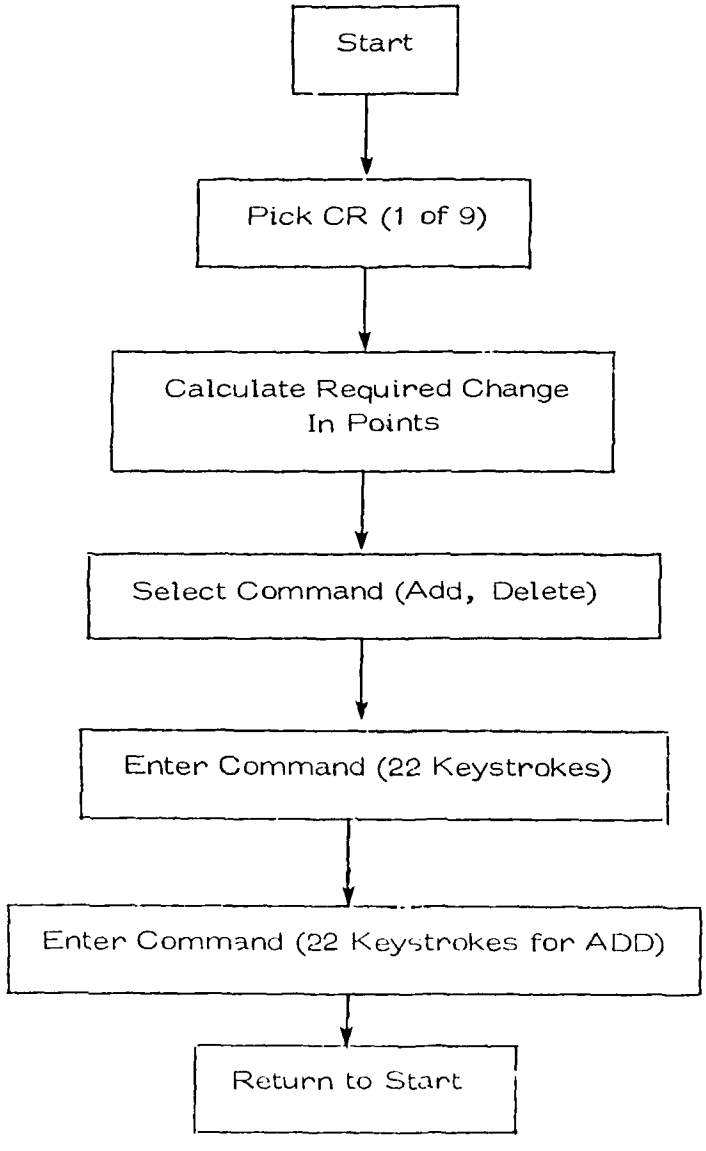
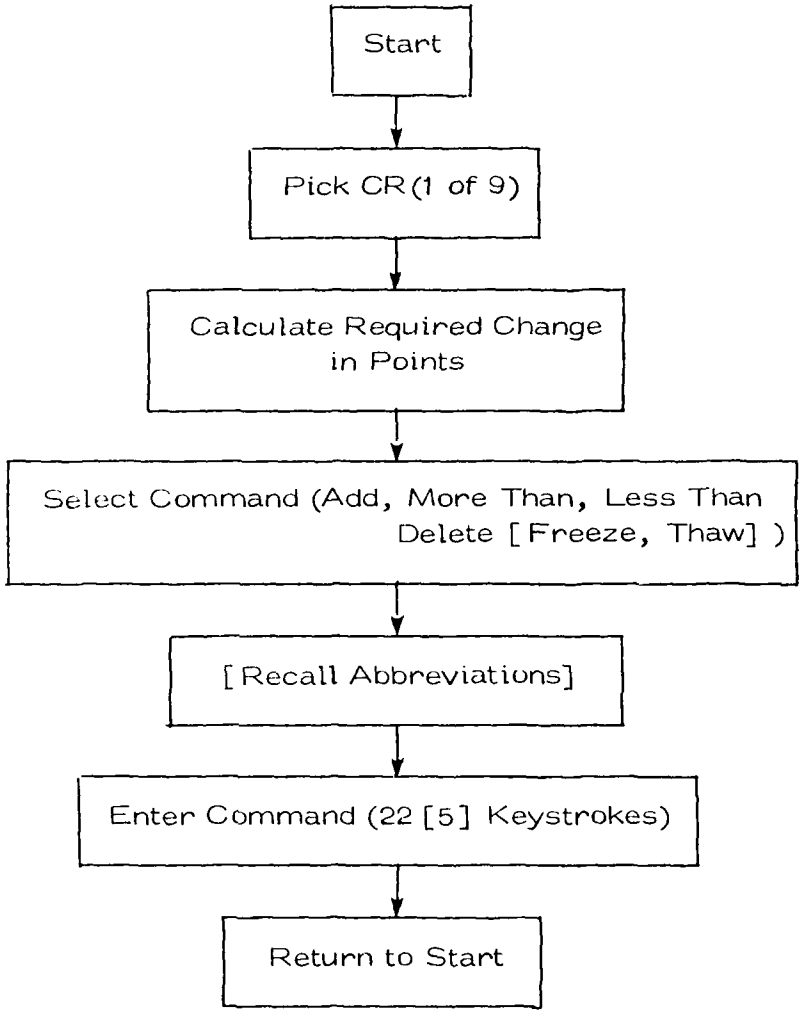
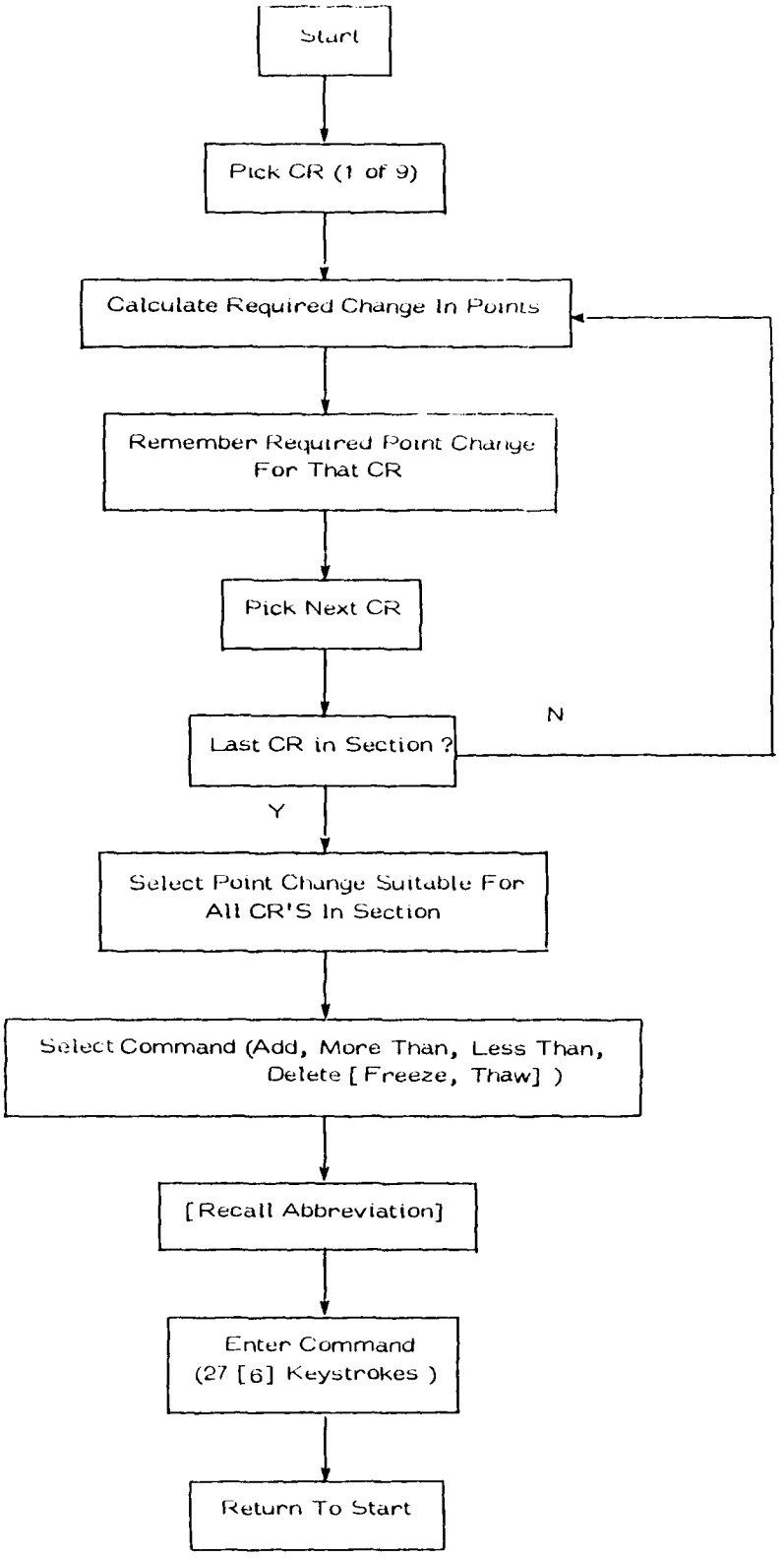


Figure 27. Cognitive Steps to Enter One ADD Command at Language Level 1



Note: [ ] indicates additional command steps and different parameter values for Language 3

Figure 28. Cognitive Steps to Enter One CR Command at Language Levels 2 and 3



Note: [ ] indicates additional command steps and different parameter values for Language 3

Figure 29. Cognitive Steps to Enter One Section Command at Language Levels 2 and 3

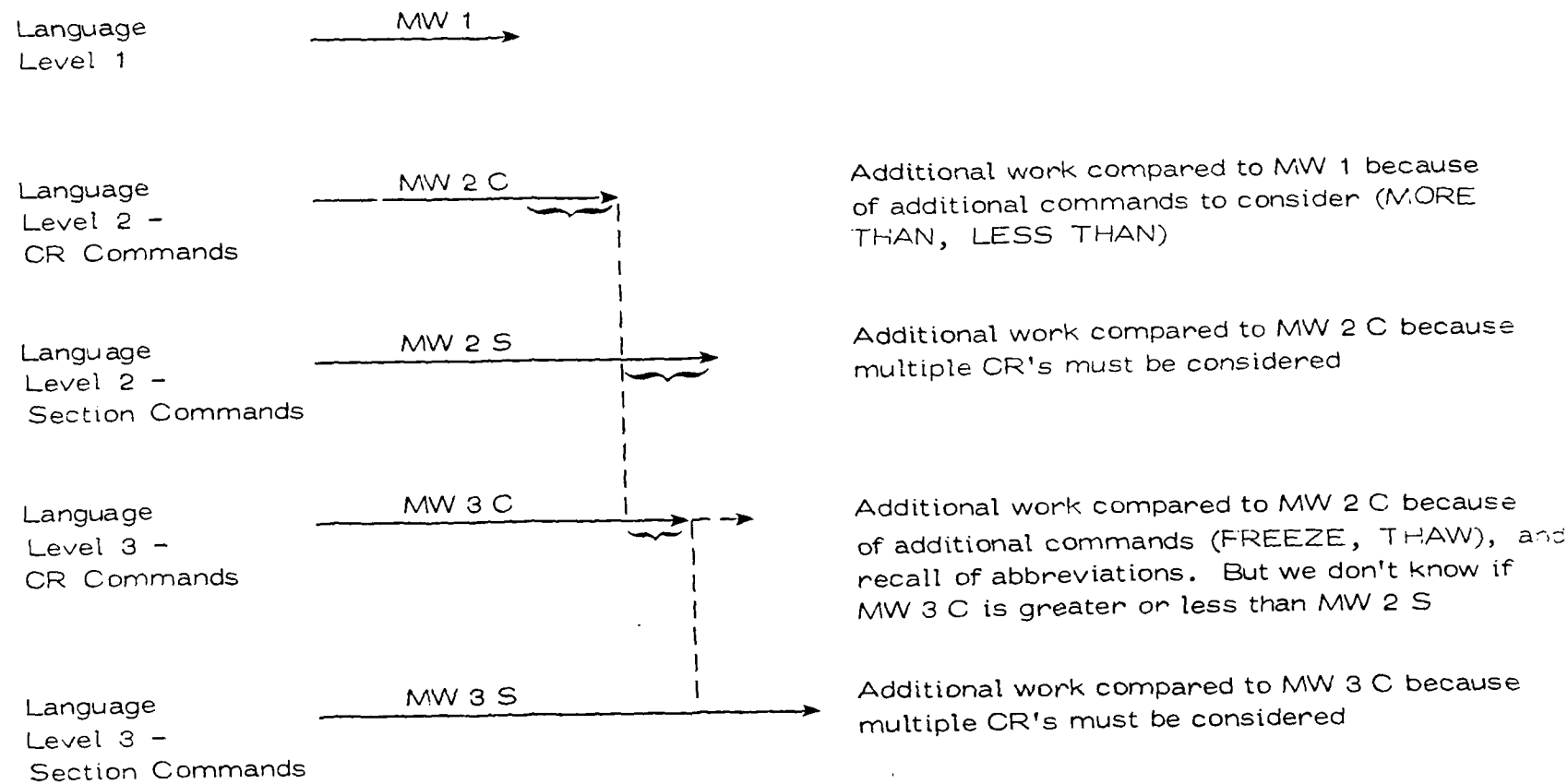


Figure 30. Comparison of Mental Work (MW) For Each Language Level

Single CR Command - Language Levels 1 and 2:

"ADD 10 POINTS TO CR 1+" Total: 22 Keystrokes

Set of three CR Commands - Language Levels 1 and 2:

"ADD 10 POINTS TO CR 1 AND
ADD 10 POINTS TO CR 2 AND
ADD 10 POINTS TO CR 3+"

Total: 74 Keystrokes

Keystroke Efficiency (3CR's) = $\frac{74}{3}$ = 24 KS/CR

Section Command - Language Level 2

"ADD 10 POINTS TO SECTION 1+"

Total: 27 Keystrokes

Keystroke Efficiency (3CR's) = $\frac{27}{3}$ = 9 KS/CR

Keystroke Efficiency (9 CR's) = $\frac{27}{9}$ = 3 KS/CR

Single CR Command - Language Level 3

"A101+" Total: 5 Keystrokes

Set of three CR Commands - Language Level 3

"A101 &A102 &A103+"

Total: 17 Keystrokes

Keystroke Efficiency (3 CR's) = $\frac{17}{3}$ = 5.66 KS/CR

Figure 31. Keystroke Model: Commands Required to Accomplish a Common Task

Section Command - Language Level 3

"A10S1+"

Total: 6 Keystrokes

Keystroke Efficiency (3CR's) = $\frac{6}{3}$ = 2 KS/CR

Keystroke Efficiency (9CR's) = $\frac{6}{9}$ = .66 KS/CR

Figure 31 (Continued). Keystroke Model: Commands Required to Accomplish a Common Task.

in efficiency could explain the participants' preference for the Section command at Language Level 2.

At Language Level 3, 5 keystrokes were required to request a test for a single CR, for an efficiency of 5 keystrokes per CR. A Section command required 6 keystrokes, so that the number of keystrokes per CR ranged from 2 to 0.66. Thus, the section commands were more efficient per CR at Language Level 3. Hence, keystroke efficiency does not explain the participants' preferred choice of CR commands at Language Level 3.

Using the terms defined in Figure 30, the command preference of the participants can be written in equation form as follows:

$$W74KS + 3MW2C \underset{P}{>} W27KS + MW2S. \quad (6)$$

This equation is read "the work (W) required to key 74 keystrokes plus three times the mental work required to use CR commands at Language Level 2 is perceived to be greater ($>_P$) than the work (W) required to key 27 keystrokes plus the mental work required to use a Section command at Language Level 2." The command preference observed at Language Level 3 can be written as follows:

$$W17KS + 3MW3C \underset{P}{<} W6KS + MW3S \quad (7)$$

We can now test two contradictory assumtions at Language Levels 2 and 3. One assumption is that the perceived total work required to use multiple (three) CR commands was greater than the perceived total work required to use one Section command. The second assumption is the reverse perception, that the perceived total work was less.

Assumption 1: Analysing 3 CR's one at a time was perceived to be more difficult than analysing multiple (3) CR's (i.e., then using the Section command). This assumption may be stated in equation form as follows, where WK_1 is some positive, but unknown, perceived work increment for Language Level 1:

$$3MW2C = MW2S + WK_2 \text{ (Language Level 2)} \quad (8)$$

$$3MW3C = MW3S + WK_3 \text{ (Language Level 3)} \quad (9)$$

Substitution into equations 6 and 7 yields, respectively:

$$W74KS + MW2S + WK_2 \underset{P}{>} W27KS + MW2S \quad (10)$$

and

$$W17KS + MW3S + WK_3 \underset{P}{<} W6KS + MW3S \quad (11)$$

Assuming further that equal entities can be subtracted from both sides of the inequalities, these equations become:

$$W47KS + WK_2 \underset{P}{>} 0 \quad (12)$$

$$W11KS + WK_3 \underset{P}{<} 0 \quad (13)$$

If work is always represented by a positive value, the second of the above two relations cannot be true. As a direct consequence, assumption 1 must be false with Language Level 3.

Assumption 2: Analysis of multiple (3) CR's was perceived to be more difficult than analysing 3 CR's one at a time. The assumption may be written in equation form as follows, where again WK_i is same positive, but unknown work increment:

$$3MW2C + WK_2 = MW2S \text{ (Language Level 2)} \quad (14)$$

$$3MW3C + WK_3 = MW3S \text{ (Language Level 3)} \quad (15)$$

Substituting into equations 6 and 7 yields, respectively:

$$W74KS + 3MW2C \underset{P}{>} W27KS + 3MW2C + WK_2 \quad (16)$$

$$W17KS + 3MW3C \underset{P}{<} W6KS + 3MW3C + WK_3 \quad (17)$$

Again, subtracting common entities from both sides of the inequality yields, respectively:

$$W47KS \underset{P}{>} WK_2 \quad (18)$$

$$W11KS \underset{P}{<} WK_3 \quad (19)$$

If WK_2 and WK_3 are assumed to represent roughly the same perception of mental work, WK, we can write:

$$W47KS > WK > W11KS \quad (20)$$

One interpretation of this inequality is that the perceived additional mental work required to use a Section command, i.e., to analyse multiple CR's, was equivalent to the work required to enter X number of keystrokes, where the value of X was greater than 11 but less than 47. Put in a different way, the reason that the participants used the Section command at Language Level 2 was that the additional mental work required to use a Section command was more than compensated for by the reduction of the manual work in typing fewer keystrokes (74 compared to 27 keystrokes). In contrast, at Language Level 3, the additional perceived mental work required to use the Section command was not sufficiently compensated for by the reduction in keystrokes (17 compared to 6). Apparently, the Language Level 3 design favored the use of CR commands.

CONCLUSIONS

The results indicate that, as additional commands, even powerful, collective commands, are added to a language (when that language is to be used for a relatively short length of time, such as when creating software-requirements specifications), the effect may be to increase the time required to input all commands. Another effect may be to increase the time taken to detect errors.

The results also indicate that abbreviations do not always reduce the amount of time required to enter a command by an amount proportional to the reduction in the number of characters. The data obtained in this experiment suggest that the time between keystrokes (and, cumulatively, the time to enter a command) can be affected by many factors, not just the length of the command. These factors appear to include whether the command is first on a line, the number of commands in the language, the number of times a command is used, plus the time to recall abbreviations, as well as the time to key in the command. The change in command entry-time for abbreviations compared to unabbreviated commands is only partly a function of the number of characters in the commands. Thus, the time required to enter a short command, such as "ADD", actually increases when the abbreviation "A" is used. The entry-time for a medium length command, of 8 to 12 characters, does not change appreciably when an abbreviation is used -- suggesting that the time needed to recall an abbreviation, at least in the problem studied here, is about the same as the amount of time saved by not having to key in a full 8-12 characters. Finally, longer commands, 14-16 characters in length (and presumably longer ones also) do realize a reduction in the entry-time when abbreviations are used.

Investigation of the different strategies employed by participants indicated a tendency to select commands by considering the perceived total work required to use a command. If a language makes available both simple commands and more powerful, high-level commands, the command actually employed by the user is the one that required the least perceived total amount of work (manual plus mental work).

The strategy analysis method identified multiple procedures or achieving a common result, and consequently, selection of one rocedure over the others by the participant implied a preference of iat procedure. Analyses of the preferred strategy permitted identification of the manual work equivalent (typing a certain number of haracters on the keyboard) to the perceived mental work required for ie procedure. Thus, the analysis method permits translations of nultiple types of work (mental and manual work) into a common manual /ork scale. This knowledge permits design tradeoffs of languages (or iore generally interfaces) to reduce the total work as perceived by he user.

The type of communication studied here, i.e., a short term ommunication between two individuals expert in different fields, is but ine of the many possible types of communications. However, it is ielieved that the results regarding the limited value of abbreviations an be applied also to report/paper readers when it is expected that it least some readers will not have previously committed to memory he abbreviations used in the documents. With modern computer echnology it is possible adaptively present to the user, via terminal screen, some or all terms in unabbreviated form instead of abbreviated orm - depending on the user's needs. With this reader-adaptive feature, he text would be stored in electronic memory with all the abbreviations is specified by the writer. Also stored is a table of abbreviations paired vith the full unabbreviated text. The reader can command presentation of full text for any abbreviation desired - the computer processor would nsert the text and then automatically justify the right and left hand nargins for presentation clarity.

REFERENCES

Anderson, V. L., & McLean, R. A. Design of experiments. New York: Marcel Dekker, Inc., 1974.

Boehm, B. W. Software engineering. IEEE Transactions on Computers. Vol. C-25, No. 12, pp. 1226-41, December 1976.

Connelly, E. M. Empirical investigation of aids for non-programming users in developing cost-effective requirements specifications Arlington, VA: Office of Naval Research, Engineering Psychology Group, 1984, (AD Number A140 149).

Jackson, M. A. Principles of program design. New York: Academic Press, 1975.

Miller, L. A. Behavioral studies of the programming process (NR 197-020). Office of Naval Research, October 1978.

Orr, K. & Associates, Inc. Structured requirements definition. Topeka, Kansas, 1981.

Ramamoorthy, C. V., and So, H. H. Software requirements and specifications: status and perspectives. Presented at the IEEE Computer Society's Second International Computer Software and Applications Conference, Chicago, Illinois, 1978.

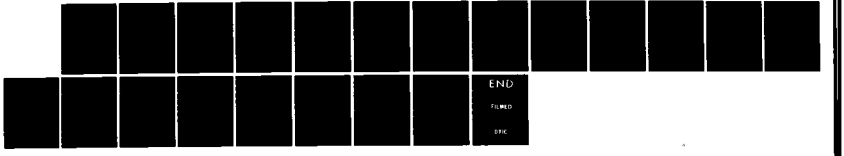
Ross, D. T., & Schoman, K. E., Jr. Structured analysis for requirements definition. IEEE Transactions on Software Engineering, 1977.

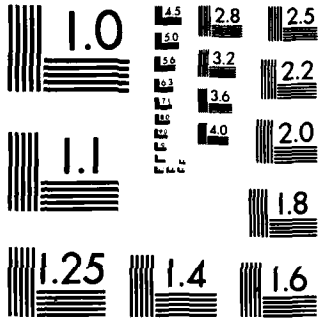
Sokal, R. R., & Rohof, R. J. Biometry. San Francisco: W. H. Freeman & Company, 1969.

Teichroew, D., & Hershey, E. A. III. PSL/SPA: a computer-aided technique for structured documentation and analysis of information processing systems. Presented at the IEEE Computer Society's Third International Computer Software & Applications Conference, Chicago, Illinois, 1979.

AD-A155 173 THE EFFECT OF COMMUNICATION-LANGUAGE DESIGNS ON SOFTWARE SPECIFICATIONS(U) PERFORMANCE MEASUREMENT ASSOCIATES INC VIENNA VA E M CONNELLY ET AL. APR 85 2/2

UNCLASSIFIED PMA-85-1 N00014-82-C-0499 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

REFERENCES (CONTINUED)

Warnier, J. D. Logical construction of systems. New York: Van Nostrand Reinhold, 1981.

Weeks, G. D., & Chapanis, A. Cooperative versus conflictive problem solving in three telecommunication modes (NR 196-135). Office of Naval Research, 1975.

Winer, B. J. Statistical principles in experimental design (2nd ed. pp. 727-736). New York: McGraw-Hill, 1971.

Yourdon, E., & Constantine, L. L. Structured design. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1979.

APPENDIX A

Table A1

Average Number of Command Lines For Total Problem

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	.0032824	29			
C	.0007334	2	.0003667	3.884	NS
Sub Within Group	.0025490	27	.0000944		
Within Subjects	.0041960	60			
A - Language	.0000493	2	.0000246	.807	NS
B - Complexity	.0023066	2	.0012033	39.449	.001
AB	.0000929	2	.0000464	1.523	NS
Error (Within)	.0016472	54	.0000305		

Table A-2

Average Number of Command Lines For Transiting

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	.0029182	29			
C	.0004545	2	.0002272	2.490	NS
Sub Within Group	.0024637	27	.0000912		
Within Subjects	.0061433	60			
A - Language	.0000214	2	.0000107	.371	NS
B - Complexity	.0045425	2	.0022712	78.819	.001
AB	.0000234	2	.0000117	.406	NS
Error (Within)	.0015561	54	.0000288		

Table A-3

Average Number of Command Lines For Homing

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	.0004193	29			
C	.0000508	2	.0000254	1.862	NS
Sub Within Group	.0003685	27	.0000136		
Within Subjects	.0014120	60			
A - Language	.0000256	2	.0000128	.674	NS
B - Complexity	.0003372	2	.0001686	8.875	.001
AB	.0000235	2	.0000117	.618	NS
Error (Within)	.0010257	54	.0000190		

Table A-4

Average Number of CR's For Total Problem

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	.0039371	29			
C	.0006049	2	.0003025	2.451	NS
Sub Within Group	.0033322	27	.0001234		
Within Subjects	.0257813	60			
A - Language	.0142220	2	.0071110	61.037	.001
B - Complexity	.0042534	2	.0021267	18.254	.001
AB	.0010148	2	.0005074	4.355	NS
Error (Within)	.0062912	54	.0001165		

Table A-5

Average Number of Times the Term "Points" is Used for Total Problem

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	.0040091	29			
C	.0000715	2	.0000357	.245	NS
Sub Within Group	.0039376	27	.0001458		
Within Subjects	.0156733	60			
A - Language	.0033482	2	.0016741	16.015	.001
B - Complexity	.0064713	2	.0032356	30.952	.001
AB	.0002089	2	.0001044	.999	NS
Error (Within)	.0056450	54	.0001045		

Table A-6

Average Number of Keystrokes Per Line For Total Problem

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	.0037081	29			
C	.0002834	2	.0001417	1.117	NS
Sub Within Group	.0034247	27	.0001268		
Within Subjects	.0207094	60			
A - Language	.0175314	2	.0087657	157.686	.001
B - Complexity	.0000139	2	.0000070	.125	NS
AB	.0001622	2	.0000811	1.459	NS
Error (Within)	.0030018	54	.0000556		

Table A-7

Average Number of Keystrokes For Total Problem

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	1.4695930	29			
C	.2466698	2	.1233349	2.723	.NS
Sub Within Group	1.2229233	27	.0452935		
Within Subjects	14.0975380	60			
A - Language	9.0003433	2	4.5001717	124.514	.001
B - Complexity	2.4088211	2	1.2044106	33.325	.001
AB	.7367134	2	.3683567	10.192	.001
Error (Within)	1.9516602	54	.0361419		

Table A-8

Average Number of Keystrokes For Transiting

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	1.6410904	29			
C	.4647427	2	.2323713	5.333	NS
Sub Within Group	1.1763477	27	.0435684		
Within Subjects	13.8654633	60			
A - Language	7.3343773	2	3.6671886	104.267	.001
B - Complexity	3.7734795	2	1.8867397	53.645	.001
AB	.8583660	2	.4291830	12.203	.001
Error (Within)	1.8992405	54	.0351711		

Table A-9

Average Number of Keystrokes For Homing

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	.4010060	29			
C	.0532738	2	.0266369	2.068	NS
Sub Within Group	.3477322	27	.0128790		
Within Subjects	.8868172	60			
A - Language	.0852656	2	.0426328	3.636	NS
B - Complexity	.1637272	2	.0818636	6.982	.001
AB	.0046508	2	.0023254	.198	NS
Error (Within)	.6331735	54	.0117254		

Table A-10

Average Time For Total Problem

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	184.9636230	29			
C	3.7238770	2	1.8619385	.277	NS
Sub Within Group	181.2397461	27	6.7125831		
Within Subjects	328.7778320	60			
A- Language	12.1022949	2	6.0511475	2.254	NS
B- Complexity	156.4719238	2	78.2359619	29.142	.001
AB	15.2338867	2	7.6169434	2.837	NS
Error (Within)	144.9697266	54	2.6846247		

Table A-11

Average Time For Transiting

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	154.8234863	29			
C	.1630859	2	.0815430	.014	NS
Sub Within Group	154.6604004	27	5.7281628		
Within Subjects	459.9255371	60			
A - Language	11.3310547	2	5.6655273	2.646	NS
B - Complexity	325.0327148	2	162.5163574	75.899	.001
AB	7.9362793	2	3.9681396	1.853	NS
Error (Within)	115.6254883	54	2.1412127		

Table A-12

Average Time For Homing

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	51.5635300	29			
C	3.2437744	2	1.6218872	.906	NS
Sub Within Group	48.3197556	27	1.7896205		
Within Subjects	152.4419098	60			
A - Language	.3151779	2	.1575890	.071	NS
B - Complexity	30.4705658	2	15.2352829	6.832	.001
AB	1.2450409	2	.6225204	.279	NS
Error (Within)	120.4111252	54	2.2298357		

Table A-13

Average Time Per Keystroke For Total Problem

Source of Variation	SS	DF	MS	F	P≤
Between Subjects	.0026687	29			
C	.0000336	2	.0000168	.172	NS
Sub Within Group	.0026352	27	.0000976		
Within Subjects	.0179158	60			
A - Language	.0155069	2	.0077534	175.666	.001
B - Complexity	.0000158	2	.0000079	.179	NS
AB	.0000097	2	.0000048	.109	NS
Error (Within)	.0023834	54	.0000441		

DISTRIBUTION LIST

OSD

CAPT Paul R. Chatelier
Office of the Deputy Under Secretary
of Defense
OUSDRE (E&LS)
Pentagon, Room 3D129
Washington, D.C. 20301

Department of the Navy

Engineering Psychology Program
Office of Naval Research
Code 442EP
800 North Quincy Street
Arlington, VA 22217-5000

Dr. Jude Franklin
Navy Center for Applied
Research In Artifical Intelligence
Code 7510
Naval Research Laboratory
Washington, D.C. 20375-5000

CAPT P. M. Curran
Code 270
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Information Sciences Division
Code 433
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Special Assistant for Marine
Corps Matters
Code 100M
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Department of the Navy

Director
Technical Information Division
Code 2627
Naval Research Laboratory
Washington, D.C. 20375-5000

Dr. Michael Melich
Communications Sciences Division
Code 7500
Naval Research Laboratory
Washington, D.C. 23075-5000

Dr. J. S. Lawson
Naval Electronic Systems Command
NELEX-06T
Washington, D.C. 20360

Naval Training Equipment Center
ATTN: Technical Library
Orlando, FL 32813

Dr. Robert G. Smith
Office of the Chief of Naval
Operations, OP987H
Personnel Logistics Plans
Washington, D.C. 20350

Mr. John Davis
Combat Control Systems Department
Code 35
Naval Underwater Systems Center
Newport, RI 02840

Human Factors Department
Code N-71
Naval Training Equipment Center
Orlando, FL 32813

Department of the Navy

Mr. Norm Beck
Combat Control Systems Department
Code 35
Naval Underwater Systems Center
Newport, RI 02840

Human Factors Engineering
Code 441
Naval Ocean Systems Center
San Diego, CA 92152

Dean of Research Administration
Naval Postgraduate School
Monterey, CA 93940

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Dr. L. Chmura
Computer Sciences & Systems
Code 7592
Naval Research Laboratory
Washington, D.C. 20375-5000

CDR C. Hutchins
Code 55
Naval Postgraduate School
Monterey, CA 93940

Human Factors Technology Admin.
Office of Naval Technology
Code MAT 0722
800 North Quincy Street
Arlington, VA 22217-5000

CDR Tom Jones
Naval Air Systems Command
Human Factors Programs
NAVAIR 330J
Washington, D.C. 20361

Department of the Navy

Mr. Philip Andrews
Naval Sea Systems Command
NAVSEA 61R
Washington, D.C. 20362

Commander
Naval Electronics Systems Command
Human Factors Engineering Branch
Code 81323
Washington, D.C. 20360

Aircrew Systems Branch
Systems Engineering Test
Directorate
U.S. Naval Test Center
Patuxent River, MD 20670

Dr. George Moeller
Human Factors Engineering Branch
Naval Submarine Base
Submarine Medical Research Lab.
Groton, CT 06340

Dr. Robert Blanchard
Code 17
Navy Personnel Research and
Development Center
San Diego, CA 92152-6800

LT Dennis McBride
Human Factors Branch
Pacific Missile Test Center
Point Mugu, CA 93042

LCDR R. Carter
Office of Chief on Naval Operations
(OP-01B)
Washington, D.C. 20350

Dean of the Academic Departments
U.S. Naval Academy
Annapolis, MD 21402

Department of the Navy

CDR W. Moroney
Naval Air Development Center
Code 602
Warminster, PA 18974

Human Factors Branch
Code 3152
Naval Weapons Center
China Lake, CA 93555

Dr. Charles Holland
Office of Naval Research
Branch Office
London
Box 39
EPO New York 09510

Dr. Eugene E. Gloye
ONR Detachment
1030 East Green Street
Pasadena, CA 91106-2485

Dr. David Mizell
ONR Detachment
1030 Green Street
Pasadena, CA 91106-2485

Dr. Glen Allgaier
Artificial Intelligence Branch
Code 444
Naval Electronics Ocean System Ctr.
San Diego, CA 92152

Dr. Steve Sacks
Naval Electronics Systems Command
Code 61R
Washington, D.C. 20363-5100

Dr. Robert A. Fleming
Human Factors Support Group
Naval Personnel Research &
Development Ctr
1411 South Fern Street
Arlington, VA 22202

Department of the Navy

Dr. Dick Kelly
Human Factors Division, Code 17
Naval Personnel Research &
Development Center
San Diego, CA 92152-6800

Department of the Army

Dr. Edgar M. Johnson
Technical Director
U.S. Army Research Institute
Alexandria, VA 22333-5600

Technical Director
U.S. Army Human Engineering Lab
Aberdeen Proving Ground, MD 21005

Director, Organizations and Systems
Research Laboratory
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Robert M. Sasmor
Director, Basic Research
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Department of the Air Force

Dr. Kenneth R. Boff
AF AMRL/HE
Wright-Patterson AFB, OH 45433

Dr. A. Fregly
U.S. Air Force Office of
Scientific Research
Life Science Directorate, NL
Bolling Air Force Base
Washington, D.C. 20332-6448

Department of the Air Force

Mr. Charles Bates, Director
Human Engineering Division
USAF AMRL/HES
Wright-Patterson AFB, OH 45433

Dr. Earl Alluisi
Chief Scientist
AFHRL/CCN
Brooks Air Force Base, TX 78235

Dr. R. K. Dismukes
Associate Director for Life Sciences
AFSOR
Bolling AFB
Washington, D.C. 20032-6448

Foreign Addresses

Dr. A. D. Baddeley
Director, Applied Psychology Unit
Medical Research Council
15 Chaucer Road
Cambridge, CB2 2EF England

Dr. Kenneth Gardner
Applied Psychology Unit
Admiralty Marine Tech. Estab.
Teddington, Middlesex
TW11 OLN
England

Other Government Agencies

Dr. M. C. Montemerlo
Information Sciences &
Human Factors
Code RC
NASA HQS
Washington, D.C. 20546

Other Government Agencies

Defense Technical Information Center
Cameron Station, Bldg. 5
Alexandria, VA 22314

Dr. Clinton Kelly
Defense Advanced Research
Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

Other Organizations

Dr. Jesse Orlansky
Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311

Dr. J. O. Chinnis, Jr.
Decision Science Consortium, Inc.
7700 Leesburg Pike
Suite 421
Falls Church, VA 22043

Dr. Stanley Deutsch
NAS-National Research Council
(COHF)
2101 Constitution Avenue, N.W.
Washington, D.C. 20418

Dr. Deborah Boehm-Davis
Department of Psychology
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Paul E. Lehner
PAR Technology Corp
7926 Jones Branch Drive
Suite 170
McLean, VA 22102

Other Organizations

Ms. Denise Benal
Essex Corporation
333 N. Fairfax Street
Alexandria, VA 22314

Dr. Richard Pew
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02238

Dr. Douglas Towne
University of Southern California
Behavioral Technology Lab
1845 South Elena Avenue, 4th Floor
Redondo Beach, CA 90277

Dr. James P. Jenkins
Nuclear Regulatory Commission
Washington, D.C. 20555

Dr. Alan Morse
Intelligent Software Systems Inc.
160 Old Farm Road
Amherst, MA 01002

Dr. Bruce Hamill
The Johns Hopkins University
Applied Physics Lab
Laurel, MD 20707

END

FILMED

7-85

DTIC